目录

1. Java 基础部分 10
1、一个".java"源文件中是否可以包括多个类（不是内部类）？有什么限制？ 10
2、Java 有没有 goto? 10
3、说说&和&&的区别。 10
4、在 JAVA 中如何跳出当前的多重嵌套循环？ 11
5、switch 语句能否作用在 byte 上，能否作用在 long 上，能否作用在 String 上? . 11
6、short s1 = 1; s1 = s1 + 1;有什么错? short s1 = 1; s1 += 1;有什么错? 12
7、char 型变量中能不能存贮一个中文汉字?为什么? 12
8、用最有效率的方法算出 2 乘以 8 等于几? 12
9、请设计一个一百亿的计算器 12
10、使用 final 关键字修饰一个变量时，是引用不能变，还是引用的对象不能变？ 14
11、"=="和 equals 方法究竟有什么区别？ 14
12、静态变量和实例变量的区别？ 15
13、是否可以从一个 static 方法内部发出对非 static 方法的调用？ 16
14、Integer 与 int 的区别 16
15、Math.round(11.5)等於多少? Math.round(-11.5)等於多少? 17
16、下面的代码有什么不妥之处? 17
17、请说出作用域 public，private，protected，以及不写时的区别 17
18、Overload 和 Override 的区别。Overloaded 的方法是否可以改变返回值的类型? 18
19、构造器 Constructor 是否可被 override? 19
20、接口是否可继承接口?抽象类是否可实现(implements)接口?抽象类是否可继承具体类(concrete class)?抽象类中是否可以有静态的 main 方法？ 19
21、写 clone()方法时，通常都有一行代码，是什么？ 19
22、面向对象的特征有哪些方面 19
23、java 中实现多态的机制是什么？ 21
24、abstract class 和 interface 有什么区别? 21
25、abstract 的 method 是否可同时是 static,是否可同时是 native，是否可同时是 synchronized? 23
26、什么是内部类？Static Nested Class 和 Inner Class 的不同。 24

VISIT…

LANZAROTE Caliente.COM

27、内部类可以引用它的包含类的成员吗？有没有什么限制？ 26
28、Anonymous Inner Class (匿名内部类)是否可以 extends(继承)其它类，是否可以 implements(实现)interface(接口)? 27
29、super.getClass()方法调用 27
30、String 是最基本的数据类型吗? 28
31、String s = "Hello";s = s + " world!";这两行代码执行后，原始的 String 对象中的内容到底变了没有？ 28
32、是否可以继承 String 类? 29
33、String s = new String("xyz");创建了几个 String Object?二者之间有什么区别？ 29
34、String 和 StringBuffer 的区别 30
35、如何把一段逗号分割的字符串转换成一个数组? 30
36、数组有没有 length()这个方法? String 有没有 length()这个方法？ 31
37、下面这条语句一共创建了多少个对象：String s="a"+"b"+"c"+"d"; 31
38、try {}里有一个 return 语句，那么紧跟在这个 try 后的 finally {}里的 code 会不会被执行，什么时候被执行，在 return 前还是后? 31
39、下面的程序代码输出的结果是多少？ 32
40、final, finally, finalize 的区别。 35
41、运行时异常与一般异常有何异同？ 35
42、error 和 exception 有什么区别? 35
43、Java 中的异常处理机制的简单原理和应用。 35
44、请写出你最常见到的 5 个 runtime exception。 36
45、JAVA 语言如何进行异常处理，关键字：throws,throw,try,catch,finally 分别代表什么意义？在 try 块中可以抛出异常吗？ 36
46、java 中有几种方法可以实现一个线程？用什么关键字修饰同步方法? stop()和 suspend()方法为何不推荐使用？ 36
47、sleep()和 wait()有什么区别? 38
48、同步和异步有何异同，在什么情况下分别使用他们？举例说明。 41
49. 下面两个方法同步吗？（自己发明） 41
50、多线程有几种实现方法?同步有几种实现方法? 42
51、启动一个线程是用 run()还是 start()? 42
52、当一个线程进入一个对象的一个 synchronized 方法后，其它线程是否可进入此对象的其它方法? 42

53、线程的基本概念、线程的基本状态以及状态之间的关系 43

54、简述 synchronized 和 java.util.concurrent.locks.Lock 的异同？ 43

55、设计 4 个线程，其中两个线程每次对 j 增加 1，另外两个线程对 j 每次减少 1。写出程序。 46

56、子线程循环 10 次，接着主线程循环 100，接着又回到子线程循环 10 次，接着再回到主线程又循环 100，如此循环 50 次，请写出程序。 48

57、介绍 Collection 框架的结构 56

58、Collection 框架中实现比较要实现什么接口 56

59、ArrayList 和 Vector 的区别 56

60、HashMap 和 Hashtable 的区别... 57

61、List 和 Map 区别? 57

62、List, Set, Map 是否继承自 Collection 接口?.. 58

63、List、Map、Set 三个接口，存取元素时，各有什么特点？ 58

64、说出 ArrayList,Vector, LinkedList 的存储性能和特性 59

65、去掉一个 Vector 集合中重复的元素 59

66、Collection 和 Collections 的区别。 59

67、Set 里的元素是不能重复的，那么用什么方法来区分重复与否呢?是用==还是 equals()?它们有何区别? 60

68、你所知道的集合类都有哪些？主要方法？ 60

69、两个对象值相同(x.equals(y) == true)，但却可有不同的 hash code，这句话对不对?... 61

70、TreeSet 里面放对象，如果同时放入了父类和子类的实例对象，那比较时使用的是父类的 compareTo 方法，还是使用的子类的 compareTo 方法，还是抛异常！ . 61

71、说出一些常用的类，包，接口，请各举 5 个.. 63

72、java 中有几种类型的流？JDK 为每种类型的流提供了一些抽象类以供继承，请说出他们分别是哪些类？ 63

73、字节流与字符流的区别... 63

74、什么是 java 序列化，如何实现 java 序列化？或者请解释 Serializable 接口的作用。 65

75、描述一下 JVM 加载 class 文件的原理机制? 66

76、heap 和 stack 有什么区别。 66

77、GC 是什么?为什么要有 GC? 66

78、垃圾回收的优点和原理。并考虑 2 种回收机制。 66

79、垃圾回收器的基本原理是什么？垃圾回收器可以马上回收内存吗？有什么办法主动通知虚拟机进行垃圾回收？ 66
80、什么时候用 assert。 67
81、java 中会存在内存泄漏吗，请简单描述。 68
82、能不能自己写个类，也叫 java.lang.String？ 72
83. Java 代码查错 73
二.算法与编程 77
1、编写一个程序，将 a.txt 文件中的单词与 b.txt 文件中的单词交替合并到 c.txt 文件中，a.txt 文件中的单词用回车符分隔，b.txt 文件中用回车或空格进行分隔。 77
2、编写一个程序，将 d:\java 目录下的所有.java 文件复制到 d:\jad 目录下，并将原来文件的扩展名从.java 改为.jad。 79
3、编写一个截取字符串的函数，输入为一个字符串和字节数，输出为按字节截取的字符串，但要保证汉字不被截取半个，如“我 ABC”，4，应该截取“我 AB”，输入“我 ABC 汉 DEF”，6，应该输出“我 ABC”，而不是“我 ABC+汉的半个”。 82
4、有一个字符串，其中包含中文字符、英文字符和数字字符，请统计和打印出各个字符的个数。 83
5、说明生活中遇到的二叉树，用 java 实现二叉树 84
6、从类似如下的文本文件中读取出所有的姓名，并打印出重复的姓名和重复的次数，并按重复次数排序： 91
7、写一个 Singleton 出来。 96
8、递归算法题 1 99
9、递归算法题 2 100
10、排序都有哪几种方法？请列举。用 JAVA 实现一个快速排序。 100
11、有数组 a[n]，用 java 代码将数组元素顺序颠倒 102
12．金额转换，阿拉伯数字的金额转换成中国传统的形式如：（￥1011）－>（一千零一拾一元整）输出。 103
三. html&JavaScript&ajax 部分 104
1. 判断第二个日期比第一个日期大 104
2. 用 table 显示 n 条记录，每 3 行换一次颜色，即 1，2，3 用红色字体，4，5，6 用绿色字体，7，8，9 用红颜色字体。 106
3、HTML 的 form 提交之前如何验证数值文本框的内容全部为数字?否则的话提示用户并终止提交? 107
4、请写出用于校验 HTML 文本框中输入的内容全部为数字的 javascript 代码 107
5、说说你用过那些 ajax 技术和框架，说说它们的区别 108

四. Java web 部分 108
1、Tomcat 的优化经验 108
2、HTTP 请求的 GET 与 POST 方式的区别 108
3、解释一下什么是 servlet; 108
4、说一说 Servlet 的生命周期? 109
5、Servlet 的基本架构 109
6、SERVLET API 中 forward()与 redirect()的区别？ 109
7、什么情况下调用 doGet()和 doPost()？ 109
8、Request 对象的主要方法： 110
9、forward 和 redirect 的区别 110
10、request.getAttribute()和 request.getParameter()有何区别? 111
11. jsp 有哪些内置对象?作用分别是什么?分别有什么方法？ 111
12. jsp 有哪些动作?作用分别是什么? 112
13、JSP 的常用指令 112
14. JSP 中动态 INCLUDE 与静态 INCLUDE 的区别？ 112
15、两种跳转方式分别是什么?有什么区别? 113
16、页面间对象传递的方法 113
17、JSP 和 Servlet 有哪些相同点和不同点，他们之间的联系是什么？ 113
18、MVC 的各个部分都有那些技术来实现?如何实现? 113
19、我们在 web 应用开发过程中经常遇到输出某种编码的字符，如 iso8859-1 等，如何输出一个某种编码的字符串？ 113
20、现在输入 n 个数字，以逗号，分开；然后可选择升或者降序排序；按提交键就在另一页面显示按什么排序，结果为，提供 reset 114
五.数据库部分 114
1、用两种方式根据部门号从高到低，工资从低到高列出每个员工的信息。 114
2、列出各个部门中工资高于本部门的平均工资的员工数和部门号，并按部门号排序 114
3、存储过程与触发器必须讲，经常被面试到? 116
4、数据库三范式是什么? 118
5、说出一些数据库优化方面的经验? 119
6、union 和 union all 有什么不同? 121
7.分页语句 122

8.用一条 SQL 语句查询出每门课都大于 80 分的学生姓名 126
9.所有部门之间的比赛组合 127
10.每个月份的发生额都比 101 科目多的科目 127
11.统计每年每月的信息 129
12.显示文章标题，发帖人、最后回复时间 131
13.删除除了 id 号不同,其他都相同的学生冗余信息 131
14.航空网的几个航班查询题： 132
15.查出比经理薪水还高的员工信息： 134
16、求出小于 45 岁的各个老师所带的大于 12 岁的学生人数 135
17.求出发帖最多的人： 136
18、一个用户表中有一个积分字段，假如数据库中有 100 多万个用户，若要在每年第一天凌晨将积分清零，你将考虑什么，你将想什么办法解决? 136
19、一个用户具有多个角色，请查询出该表中具有该用户的所有角色的其他用户。 137
20. xxx 公司的 sql 面试 138
21、注册 Jdbc 驱动程序的三种方式 139
22、用 JDBC 如何调用存储过程 139
23、JDBC 中的 PreparedStatement 相比 Statement 的好处 141
24. 写一个用 jdbc 连接并访问 oracle 数据的程序代码 141
25、Class.forName 的作用?为什么要用? 141
26、大数据量下的分页解决方法。 141
27、用 JDBC 查询学生成绩单,把主要代码写出来（考试概率极大）. 142
28、这段代码有什么不足之处? 143
29、说出数据连接池的工作机制是什么? 143
30、为什么要用 ORM? 和 JDBC 有何不一样? 144
六. XML 部分 144
1、xml 有哪些解析技术?区别是什么? 144
2、你在项目中用到了 xml 技术的哪些方面?如何实现的? 144
3、用 jdom 解析 xml 文件时如何解决中文问题?如何解析? 144
4、编程用 JAVA 解析 XML 的方式 145
5、XML 文档定义有几种形式？它们之间有何本质区别？解析 XML 文档有哪几种方式？ 148

七.流行的框架与新技术 148
1、谈谈你对 Struts 的理解。 149
2、谈谈你对 Hibernate 的理解。 149
3、AOP 的作用。 150
4、你对 Spring 的理解。 150
5、谈谈 Struts 中的 Action servlet。 151
6、Struts 优缺点 优点： 1. 实现 MVC 模式，结构清晰,使开发者只关注业务逻辑的实现. 151
7、STRUTS 的应用(如 STRUTS 架构) 153
8、说说 struts1 与 struts2 的区别。 153
9、hibernate 中的 update()和 saveOrUpdate()的区别，session 的 load()和 get()的区别。 154
10、简述 Hibernate 和 JDBC 的优缺点?如何书写一个 one to many 配置文件. 154
11、iBatis 与 Hibernate 有什么不同? 154
12、写 Hibernate 的一对多和多对一双向关联的 orm 配置? 154
12.1、hibernate 的 inverse 属性的作用? 154
13、在 DAO 中如何体现 DAO 设计模式? 155
14、spring+Hibernate 中委托方案怎么配置? 155
15、spring+Hibernate 中委托方案怎么配置? 155
16. hibernate 进行多表查询每个表中各取几个字段，也就是说查询出来的结果集没有一个实体类与之对应如何解决； 155
17.介绍一下 Hibernate 的二级缓存 155
18、Spring 的依赖注入是什么意思?给一个 Bean 的 message 属性,字符串类型,注入值为"Hello"的 XML 配置文件该怎么写? 158
19、Jdo 是什么? 158
20、什么是 spring 的 IOC AOP 158
21、STRUTS 的工作流程！ 158
22、spring 与 EJB 的区别！！ 158
八.软件工程与设计模式 158
1、UML 方面 158
2、j2ee 常用的设计模式？说明工厂模式。 158
3、开发中都用到了那些设计模式?用在什么场合? 159
九. j2ee 部分 159

1、BS 与 CS 的联系与区别。 159
1．硬件环境不同: 159
2．对安全要求不同 160
3．对程序架构不同 160
4．软件重用不同 160
5．系统维护不同 160
6．处理问题不同 160
7．用户接口不同 160
2、应用服务器与 WEB SERVER 的区别？ 161
3、应用服务器有那些？ 161
4、J2EE 是什么？ 161
5、J2EE 是技术还是平台还是框架？什么是 J2EE 161
6、请对以下在 J2EE 中常用的名词进行解释(或简单描述) 161
7、如何给 weblogic 指定大小的内存? 162
8、如何设定的 weblogic 的热启动模式(开发模式)与产品发布模式? 162
9、如何启动时不需输入用户名与密码? 162
10、在 weblogic 管理制台中对一个应用域(或者说是一个网站,Domain)进行 jms 及 ejb 或连接池等相关信息进行配置后,实际保存在什么文件中? 162
11、说说 weblogic 中一个 Domain 的缺省目录结构?比如要将一个简单的 helloWorld.jsp 放入何目录下,然的在浏览器上就可打入 http://主机:端口号//helloword.jsp 就可以看到运行结果了?又比如这其中用到了一个自己写的 javaBean 该如何办? 162
12、在 weblogic 中发布 ejb 需涉及到哪些配置文件 163
13、如何在 weblogic 中进行 ssl 配置与客户端的认证配置或说说 j2ee(标准)进行 ssl 的配置? 163
14、如何查看在 weblogic 中已经发布的 EJB? 163
十. EJB 部分 163
1、EJB 是基于哪些技术实现的？并说出 SessionBean 和 EntityBean 的区别，StatefulBean 和 StatelessBean 的区别。 163
2、简要讲一下 EJB 的 7 个 Transaction Level? 164
3、EJB 与 JAVA BEAN 的区别？ 164
4、EJB 包括（SessionBean,EntityBean）说出他们的生命周期，及如何管理事务的？ 164

5、EJB 容器提供的服务.. 164
6、EJB 的激活机制 164
7、EJB 的几种类型 165
8、客服端调用 EJB 对象的几个基本步骤... 165
十一. webservice 部分 165
1、WEB SERVICE 名词解释。JSWDL 开发包的介绍。JAXP、JAXM 的解释。SOAP、UDDI,WSDL 解释。 165
2、CORBA 是什么?用途是什么? 165
3. Linux... 166
4、LINUX 下线程，GDI 类的解释。 166
5. 问得稀里糊涂的题.. 166
6、四种会话跟踪技术... 166
7、简述逻辑操作(&,|,^)与条件操作(&&,||)的区别。 166
十二.其他.. 166
1、请用英文简单介绍一下自己.. 166
2、请把 http://tomcat.apache.org/ 首页的这一段话用中文翻译一下?.................. 166
3、美资软件公司 JAVA 工程师电话面试题目 167

这套面试题主要目的是帮助那些还没有 java 软件开发实际工作经验，而正在努力寻找 java 软件开发工作的朋友在笔试时更好地赢得笔试和面试。由于这套面试题涉及的范围很泛，很广，很杂，大家不可能一天两天就看完和学完这套面试宝典，即使你已经学过了有关的技术，那么至少也需要一个月的时间才能消化和掌握这套面试宝典，所以，大家应该早作准备，从拿到这套面试宝典之日起，就要坚持在每天闲暇之余学习其中几道题目，日积月累，等到出去面试时，一切都水到渠成，面试时就自然会游刃有余了。

答题时，先答是什么，再答有什么作用和要注意什么（这部分最重要，展现自己的心得）

答案的段落分别，层次分明，条理清晰都非常重要，从这些表面的东西也可以看出一个人的习惯、办事风格、条理等。

要讲你做出答案的思路过程，或者说你记住答案的思想都写下来。把答题想着是辩论赛。答题就是给别人讲道理、摆事实。答题不局限于什么格式和形式，就是要将自己的学识展现出来！

别因为人家题目本来就模棱两可，你就心里胆怯和没底气了，不敢回答了。你要大胆地指出对方题目很模糊和你的观点，不要把面试官想得有多高，其实他和你就是差不多的，你想想，如果他把你招进去了，你们以后就是同事了，可不是差不多的吗？

关于就业薪水，如果你是应届生，那不能要高工资，好比大饼的故事，没有文凭还想拿高工资，就去中关村缺什么补什么吧！少数人基础确实很好，在校期间确实又做过一些项目，那仍然是可以要到相对高的工资的。

1. Java 基础部分

基础部分的顺序：基本语法，类相关的语法，内部类的语法，继承相关的语法，异常的语法，线程的语法，集合的语法，io 的语法，虚拟机方面的语法。

1、一个".java"源文件中是否可以包括多个类（不是内部类）？有什么限制？

可以有多个类，但只能有一个 public 的类，并且 public 的类名必须与文件名相一致。

2、Java 有没有 goto?

java 中的保留字，现在没有在 java 中使用。

3、说说&和&&的区别。

&和&&都可以用作逻辑与的运算符，表示逻辑与（and），当运算符两边的表达式的结果都为 true 时，整个运算结果才为 true，否则，只要有一方为 false，则结果为 false。

&&还具有短路的功能，即如果第一个表达式为 false，则不再计算第二个表达式，例如，对于 if(str != null&& !str.equals(“”))表达式，当 str 为 null 时，后面的表达式不会执行，所以不会出现 NullPointerException 如果将&&改为&，则会抛出 NullPointerException 异常。

If(x==33 &++y>0) y 会增长，If(x==33 && ++y>0)不会增长

&还可以用作位运算符，当&操作符两边的表达式不是 boolean 类型时，&表示按位与操作，

我们通常使用0x0f 来与一个整数进行&运算，来获取该整数的最低4个 bit 位，例如，0x31 & 0x0f 的结果为0x01。

备注：这道题先说两者的共同点，再说出&&和&的特殊之处，并列举一些经典的例子来表明自己理解透彻深入、实际经验丰富。

4、在 JAVA 中如何跳出当前的多重嵌套循环？

在 Java 中，要想跳出多重循环，可以在外面的循环语句前定义一个标号，然后在里层循环体的代码中使用带有标号的 break 语句，即可跳出外层循环。例如，

```
ok:
for(int i=0;i<10;i++)       {
        for(int j=0;j<10;j++)               {
                System.out.println("i=" + i + ",j=" + j);
                if(j == 5) break ok;
        }
}
```

另外，我个人通常并不使用标号这种方式，而是让外层的循环条件表达式的结果可以受到里层循环体代码的控制，例如，要在二维数组中查找到某个数字。

```
int arr[][] ={{1,2,3},{4,5,6,7},{9}};
boolean found = false;
for(int i=0;i<arr.length&& !found;i++)          {
        for(int j=0;j<arr[i].length;j++){
                System.out.println("i=" + i + ",j=" + j);
                if(arr[i][j]   ==5) {
                        found = true;
                        break;
                }
        }
}
```

5、switch 语句能否作用在 byte 上，能否作用在 long 上，能否作用在 String 上?

在 switch（expr1）中，expr1只能是一个整数表达式或者枚举常量（更大字体），整数表达式可以是 int 基本类型或 Integer 包装类型，由于，byte,short,char 都可以隐含转换为 int，

所以，这些类型以及这些类型的包装类型也是可以的。显然，long 和 String 类型都不符合 switch 的语法规定，并且不能被隐式转换成 int 类型，所以，它们不能作用于 swtich 语句中。

6、short s1 = 1; s1 = s1 + 1;有什么错? short s1 = 1; s1 += 1;有什么错?

对于 short s1 = 1; s1 = s1 + 1;由于 s1+1运算时会自动提升表达式的类型，所以结果是 int 型，再赋值给 short 类型 s1时，编译器将报告需要强制转换类型的错误。

对于 short s1 = 1; s1 += 1;由于 +=是 java 语言规定的运算符，java 编译器会对它进行特殊处理，因此可以正确编译。

7、char 型变量中能不能存贮一个中文汉字?为什么?

char 型变量是用来存储 Unicode 编码的字符的，unicode 编码字符集中包含了汉字，所以，char 型变量中当然可以存储汉字啦。不过，如果某个特殊的汉字没有被包含在 unicode 编码字符集中，那么，这个 char 型变量中就不能存储这个特殊汉字。补充说明：unicode 编码占用两个字节，所以，char 类型的变量也是占用两个字节。

备注：后面一部分回答虽然不是在正面回答题目，但是，为了展现自己的学识和表现自己对问题理解的透彻深入，可以回答一些相关的知识，做到知无不言，言无不尽。

8、用最有效率的方法算出 2 乘以 8 等于几?

2 << 3，

因为将一个数左移 n 位，就相当于乘以了2的 n 次方，那么，一个数乘以8只要将其左移3位即可，而位运算 cpu 直接支持的，效率最高，所以，2乘以8等於几的最效率的方法是2 << 3。

9、请设计一个一百亿的计算器

首先要明白这道题目的考查点是什么，一是大家首先要对计算机原理的底层细节要清楚、要知道加减法的位运算原理和知道计算机中的算术运算会发生越界的情况，二是要具备一定的面向对象的设计思想。

首先，计算机中用固定数量的几个字节来存储的数值，所以计算机中能够表示的数值是有一定的范围的，为了便于讲解和理解，我们先以 byte 类型的整数为例，它用1个字节进行存储，表示的最大数值范围为-128到+127。-1在内存中对应的二进制数据为11111111，如果两个-1相加，不考虑 Java 运算时的类型提升，运算后会产生进位，二进制结果为1,11111110，由于进位后超过了 byte 类型的存储空间，所以进位部分被舍弃，即最终的结果为11111110，也就是-2，这正好利用溢位的方式实现了负数的运算。-128在内存中对应的二进制数据为10000000，如果两个-128相加，不考虑 Java 运算时的类型提升，运算后会产生进位，二进制结果为1,00000000，由于进位后超过了 byte 类型的存储空间，所以进位部分被舍弃，即最终的结果为00000000，也就是0，这样的结果显然不是我们期望的，这说明计算机中的算术运算是会发生越界情况的，两个数值的运算结果不能超过计算机中的该类型的数值范

围。由于 Java 中涉及表达式运算时的类型自动提升，我们无法用 byte 类型来做演示这种问题和现象的实验，大家可以用下面一个使用整数做实验的例子程序体验一下：

```
int a = Integer.MAX_VALUE;
int b = Integer.MAX_VALUE;
int sum = a + b;
System.out.println("a="+a+",b="+b+",sum="+sum);
```

先不考虑 long 类型，由于 int 的正数范围为2的31次方，表示的最大数值约等于2*1000*1000*1000，也就是20亿的大小，所以，要实现一个一百亿的计算器，我们得自己设计一个类可以用于表示很大的整数，并且提供了与另外一个整数进行加减乘除的功能，大概功能如下：

（1）这个类内部有两个成员变量，一个表示符号，另一个用字节数组表示数值的二进制数

（2）有一个构造方法，把一个包含有多位数值的字符串转换到内部的符号和字节数组中

（3）提供加减乘除的功能

```
public class BigInteger{
	int sign;
	byte[] val;
	public Biginteger(String val) {
		sign = ;
		val = ;
	}
	public BigInteger add(BigInteger other)		{

	}
	public BigInteger subtract(BigInteger other) {

	}
	public BigInteger multiply(BigInteger other){

	}
	public BigInteger divide(BigInteger other){
```

```
        }
}
```

备注：要想写出这个类的完整代码，是非常复杂的，如果有兴趣的话，可以参看 jdk 中自带的 java.math.BigInteger 类的源码。面试的人也知道谁都不可能在短时间内写出这个类的完整代码的，他要的是你是否有这方面的概念和意识，他最重要的还是考查你的能力，所以，你不要因为自己无法写出完整的最终结果就放弃答这道题，你要做的就是你比别人写得多，证明你比别人强，你有这方面的思想意识就可以了，毕竟别人可能连题目的意思都看不懂，什么都没写，你要敢于答这道题，即使只答了一部分，那也与那些什么都不懂的人区别出来，拉开了距离，算是矮子中的高个，机会当然就属于你了。另外，答案中的框架代码也很重要，体现了一些面向对象设计的功底，特别是其中的方法命名很专业，用的英文单词很精准，这也是能力、经验、专业性、英语水平等多个方面的体现，会给人留下很好的印象，在编程能力和其他方面条件差不多的情况下，英语好除了可以使你获得更多机会外，薪水可以高出一千元。

10、使用 final 关键字修饰一个变量时，是引用不能变，还是引用的对象不能变？

使用 final 关键字修饰一个变量时，是指引用变量不能变，引用变量所指向的对象中的内容还是可以改变的。例如，对于如下语句：

```
final StringBuffer a=new StringBuffer("immutable");
```

执行如下语句将报告编译期错误：

```
a=new StringBuffer("");
```

但是，执行如下语句则可以通过编译：

```
a.append(" broken!");
```

有人在定义方法的参数时，可能想采用如下形式来阻止方法内部修改传进来的参数对象：

```
public void method(final  StringBuffer param){
}
```

实际上，这是办不到的，在该方法内部仍然可以增加如下代码来修改参数对象：

```
        param.append("a");
```

11、"=="和 equals 方法究竟有什么区别？

（单独把一个东西说清楚，然后再说清楚另一个，这样，它们的区别自然就出来了，混在一起说，则很难说清楚）

==操作符专门用来比较两个变量的值是否相等，也就是用于比较变量所对应的内存中所存储的数值是否相同，要比较两个基本类型的数据或两个引用变量是否相等，只能用==操作符。

如果一个变量指向的数据是对象类型的，那么，这时候涉及了两块内存，对象本身占用一块内存（堆内存），变量也占用一块内存，例如 Objet obj = new Object();变量 obj 是一个内存，new Object()是另一个内存，此时，变量 obj 所对应的内存中存储的数值就是对象占用的那块内存的首地址。对于指向对象类型的变量，如果要比较两个变量是否指向同一个对象，即要看这两个变量所对应的内存中的数值是否相等，这时候就需要用==操作符进行比较。

equals 方法是用于比较两个独立对象的内容是否相同，就好比去比较两个人的长相是否相同，它比较的两个对象是独立的。例如，对于下面的代码：

String a=new String("foo");

String b=new String("foo");

两条 new 语句创建了两个对象，然后用 a/b 这两个变量分别指向了其中一个对象，这是两个不同的对象，它们的首地址是不同的，即 a 和 b 中存储的数值是不相同的，所以，表达式 a==b 将返回 false，而这两个对象中的内容是相同的，所以，表达式 a.equals(b)将返回 true。

在实际开发中，我们经常要比较传递进行来的字符串内容是否等，例如，String input = …;input.equals("quit")，许多人稍不注意就使用==进行比较了，这是错误的，随便从网上找几个项目实战的教学视频看看，里面就有大量这样的错误。记住，字符串的比较基本上都是使用 equals 方法。

如果一个类没有自己定义 equals 方法，那么它将继承 Object 类的 equals 方法，Object 类的 equals 方法的实现代码如下：

```
boolean equals(Object o){
return this==o;
}
```

这说明，如果一个类没有自己定义 equals 方法，它默认的 equals 方法（从 Object 类继承的）就是使用==操作符，也是在比较两个变量指向的对象是否是同一对象，这时候使用 equals 和使用==会得到同样的结果，如果比较的是两个独立的对象则总返回 false。如果你编写的类希望能够比较该类创建的两个实例对象的内容是否相同，那么你必须覆盖 equals 方法，由你自己写代码来决定在什么情况即可认为两个对象的内容是相同的。

12、静态变量和实例变量的区别？

在语法定义上的区别：静态变量前要加 static 关键字，而实例变量前则不加。

在程序运行时的区别：实例变量属于某个对象的属性，必须创建了实例对象，其中的实例变

量才会被分配空间，才能使用这个实例变量。静态变量不属于某个实例对象，而是属于类，所以也称为类变量，只要程序加载了类的字节码，不用创建任何实例对象，静态变量就会被分配空间，静态变量就可以被使用了。总之，实例变量必须创建对象后才可以通过这个对象来使用，静态变量则可以直接使用类名来引用。

例如，对于下面的程序，无论创建多少个实例对象，永远都只分配了一个 staticVar 变量，并且每创建一个实例对象，这个 staticVar 就会加1；但是，每创建一个实例对象，就会分配一个 instanceVar，即可能分配多个 instanceVar，并且每个 instanceVar 的值都只自加了1次。

```
public class VariantTest{
        public static int staticVar = 0;
        public int instanceVar = 0;
        public VariantTest(){
                staticVar++;
                instanceVar++;
                System.out.println("staticVar=" + staticVar + ",instanceVar="+
instanceVar);
        }
}
```

备注：这个解答除了说清楚两者的区别外，最后还用一个具体的应用例子来说明两者的差异，体现了自己有很好的解说问题和设计案例的能力，思维敏捷，超过一般程序员，有写作能力！

13、是否可以从一个 static 方法内部发出对非 static 方法的调用？

不可以。因为非 static 方法是要与对象关联在一起的，必须创建一个对象后，才可以在该对象上进行方法调用，而 static 方法调用时不需要创建对象，可以直接调用。也就是说，当一个 static 方法被调用时，可能还没有创建任何实例对象，如果从一个 static 方法中发出对非 static 方法的调用，那个非 static 方法是关联到哪个对象上的呢？这个逻辑无法成立，所以，一个 static 方法内部发出对非 static 方法的调用。

14、Integer 与 int 的区别

int 是 java 提供的8种原始数据类型之一。Java 为每个原始类型提供了封装类，Integer 是 java 为 int 提供的封装类。int 的默认值为0，而 Integer 的默认值为 null，即 Integer 可以区分出未赋值和值为0的区别，int 则无法表达出未赋值的情况，例如，要想表达出没有参加考试和考试成绩为0的区别，则只能使用 Integer。在 JSP 开发中，Integer 的默认为 null，所以用 el 表达式在文本框中显示时，值为空白字符串，而 int 默认的默认值为0，所以用 el 表达式

在文本框中显示时，结果为0，所以，int 不适合作为 web 层的表单数据的类型。

在 Hibernate 中，如果将 OID 定义为 Integer 类型，那么 Hibernate 就可以根据其值是否为 null 而判断一个对象是否是临时的，如果将 OID 定义为了 int 类型，还需要在 hbm 映射文件中设置其 unsaved-value 属性为0。

另外，Integer 提供了多个与整数相关的操作方法，例如，将一个字符串转换成整数，Integer 中还定义了表示整数的最大值和最小值的常量。

15、Math.round(11.5)等於多少? Math.round(-11.5)等於多少?

Math 类中提供了三个与取整有关的方法：ceil、floor、round，这些方法的作用与它们的英文名称的含义相对应，例如，ceil 的英文意义是天花板，该方法就表示向上取整，Math.ceil(11.3)的结果为12,Math.ceil(-11.3)的结果是-11；floor 的英文意义是地板，该方法就表示向下取整，Math.ceil(11.6)的结果为11,Math.ceil(-11.6)的结果是-12；最难掌握的是 round 方法，它表示"四舍五入"，算法为 Math.floor(x+0.5)，即将原来的数字加上0.5后再向下取整，所以，Math.round(11.5)的结果为12，Math.round(-11.5)的结果为-11。

16、下面的代码有什么不妥之处?

1. if(username.equals("zxx")){}

username 可能为 NULL,会报空指针错误；改为"zxx".equals(username)

2. **int** x = 1;

return x==1?true:false; 这个改成 return x==1;就可以!

17、请说出作用域 public，private，protected，以及不写时的区别

这四个作用域的可见范围如下表所示。

说明：如果在修饰的元素上面没有写任何访问修饰符，则表示 friendly。

作用域	当前类	同一包（package）	子孙类	其他包（package）
public	√	√	√	√
protected	√	√	√	×
friendly	√	√	×	×
private	√	×	×	×

备注：只要记住了有4种访问权限，4个访问范围，然后将全选和范围在水平和垂直方向上分别按排从小到大或从大到小的顺序排列，就很容易画出上面的图了。

18、Overload 和 Override 的区别。Overloaded 的方法是否可以改变返回值的类型?

Overload 是重载的意思，**Override** 是覆盖的意思，也就是重写。

重载 Overload 表示同一个类中可以有多个名称相同的方法，但这些方法的参数列表各不相同（即参数个数或类型不同）。

重写 Override 表示子类中的方法可以与父类中的某个方法的名称和参数完全相同，通过子类创建的实例对象调用这个方法时，将调用子类中的定义方法，这相当于把父类中定义的那个完全相同的方法给覆盖了，这也是面向对象编程的多态性的一种表现。子类覆盖父类的方法时，只能比父类抛出更少的异常，或者是抛出父类抛出的异常的子异常，因为子类可以解决父类的一些问题，不能比父类有更多的问题。子类方法的访问权限只能比父类的更大，不能更小。如果父类的方法是 private 类型，那么，子类则不存在覆盖的限制，相当于子类中增加了一个全新的方法。

至于 Overloaded 的方法是否可以改变返回值的类型这个问题，要看你倒底想问什么呢？这个题目很模糊。如果几个 Overloaded 的方法的参数列表不一样，它们的返回者类型当然也可以不一样。但我估计你想问的问题是：如果两个方法的参数列表完全一样，是否可以让它们的返回值不同来实现重载 Overload。这是不行的，我们可以用反证法来说明这个问题，因为我们有时候调用一个方法时也可以不定义返回结果变量，即不要关心其返回结果，例如，我们调用 map.remove(key)方法时，虽然 remove 方法有返回值，但是我们通常都不会定义接收返回结果的变量，这时候假设该类中有两个名称和参数列表完全相同的方法，仅仅是返回类型不同，java 就无法确定编程者倒底是想调用哪个方法了，因为它无法通过返回结果类型来判断。

override 可以翻译为覆盖，从字面就可以知道，它是覆盖了一个方法并且对其重写，以求达到不同的作用。对我们来说最熟悉的覆盖就是对接口方法的实现，在接口中一般只是对方法进行了声明，而我们在实现时，就需要实现接口声明的所有方法。除了这个典型的用法以外，我们在继承中也可能会在子类覆盖父类中的方法。在覆盖要注意以下的几点：

1、覆盖的方法的标志必须要和被覆盖的方法的标志完全匹配，才能达到覆盖的效果；

2、覆盖的方法的返回值必须和被覆盖的方法的返回一致；

3、覆盖的方法所抛出的异常必须和被覆盖方法的所抛出的异常一致，或者是其子类；

4、被覆盖的方法不能为 private，否则在其子类中只是新定义了一个方法，并没有对其进行覆盖。

overload 对我们来说可能比较熟悉，可以翻译为重载，它是指我们可以定义一些名称相同的方法，通过定义不同的输入参数来区分这些方法，然后再调用时，VM 就会根据不同的参

数样式，来选择合适的方法执行。在使用重载要注意以下的几点：

1、在使用重载时只能通过不同的参数样式。例如，不同的参数类型，不同的参数个数，不同的参数顺序（当然，同一方法内的几个参数类型必须不一样，例如可以是 fun(int,float)，但是不能为 fun(int,int)）；

2、不能通过访问权限、返回类型、抛出的异常进行重载；

3、方法的异常类型和数目不会对重载造成影响；

4、对于继承来说，如果某一方法在父类中是访问权限是 priavte，那么就不能在子类对其进行重载，如果定义的话，也只是定义了一个新方法，而不会达到重载的效果。

19、构造器 Constructor 是否可被 override?

构造器 Constructor 不能被继承，因此不能重写 Override，但可以被重载 Overload。

20、接口是否可继承接口?抽象类是否可实现(implements)接口?抽象类是否可继承具体类(concrete class)?抽象类中是否可以有静态的 main 方法？

接口可以继承接口。抽象类可以实现(implements)接口，抽象类可以继承具体类。抽象类中可以有静态的 main 方法。

备注：只要明白了接口和抽象类的本质和作用，这些问题都很好回答，你想想，如果你是 java 语言的设计者，你是否会提供这样的支持，如果不提供的话，有什么理由吗？如果你没有道理不提供，那答案就是肯定的了。

只有记住抽象类与普通类的唯一区别：就是不能创建实例对象和允许有 abstract 方法。

21、写 clone()方法时，通常都有一行代码，是什么？

clone 有缺省行为，

super.clone();

因为首先要把父类中的成员复制到位，然后才是复制自己的成员。

22、面向对象的特征有哪些方面

计算机软件系统是现实生活中的业务在计算机中的映射，而现实生活中的业务其实就是一个个对象协作的过程。面向对象编程就是按现实业务一样的方式将程序代码按一个个对象进行组织和编写，让计算机系统能够识别和理解用对象方式组织和编写的程序代码，这样就可以把现实生活中的业务对象映射到计算机系统中。

面向对象的编程语言有，吗等4个主要的特征。

封装：

封装是保证软件部件具有优良的模块性的基础，封装的目标就是要实现软件部件的“高内聚、

低耦合”，防止程序相互依赖性而带来的变动影响。在面向对象的编程语言中，对象是封装的最基本单位，面向对象的封装比传统语言的封装更为清晰、更为有力。面向对象的封装就是把描述一个对象的属性和行为的代码封装在一个“模块”中，也就是一个类中，属性用变量定义，行为用方法进行定义，方法可以直接访问同一个对象中的属性。通常情况下，只要记住让变量和访问这个变量的方法放在一起，将一个类中的成员变量全部定义成私有的，只有这个类自己的方法才可以访问到这些成员变量，这就基本上实现对象的封装，就很容易找出要分配到这个类上的方法了，就基本上算是会面向对象的编程了。把握一个原则：把对同一事物进行操作的方法和相关的方法放在同一个类中，把方法和它操作的数据放在同一个类中。

例如，人要在黑板上画圆，这一共涉及三个对象：人、黑板、圆，画圆的方法要分配给哪个对象呢？由于画圆需要使用到圆心和半径，圆心和半径显然是圆的属性，如果将它们在类中定义成了私有的成员变量，那么，画圆的方法必须分配给圆，它才能访问到圆心和半径这两个属性，人以后只是调用圆的画圆方法、表示给圆发给消息而已，画圆这个方法不应该分配在人这个对象上，这就是面向对象的封装性，即将对象封装成一个高度自治和相对封闭的个体，对象状态（属性）由这个对象自己的行为（方法）来读取和改变。一个更便于理解的例子就是，司机将火车刹住了，刹车的动作是分配给司机，还是分配给火车，显然，应该分配给火车，因为司机自身是不可能有那么大的力气将一个火车给停下来的，只有火车自己才能完成这一动作，火车需要调用内部的离合器和刹车片等多个器件协作才能完成刹车这个动作，司机刹车的过程只是给火车发了一个消息，通知火车要执行刹车动作而已。

抽象：

抽象就是找出一些事物的相似和共性之处，然后将这些事物归为一个类，这个类只考虑这些事物的相似和共性之处，并且会忽略与当前主题和目标无关的那些方面，将注意力集中在与当前目标有关的方面。例如，看到一只蚂蚁和大象，你能够想象出它们的相同之处，那就是抽象。抽象包括行为抽象和状态抽象两个方面。例如，定义一个 Person 类，如下：

```
classPerson{
        String name;
        int age;
}
```

人本来是很复杂的事物，有很多方面，但因为当前系统只需要了解人的姓名和年龄，所以上面定义的类中只包含姓名和年龄这两个属性，这就是一种抽像，使用抽象可以避免考虑一些与目标无关的细节。我对抽象的理解就是不要用显微镜去看一个事物的所有方面，这样涉及的内容就太多了，而是要善于划分问题的边界，当前系统需要什么，就只考虑什么。

继承：

在定义和实现一个类的时候，可以在一个已经存在的类的基础之上来进行，把这个已经存在的类所定义的内容作为自己的内容，并可以加入若干新的内容，或修改原来的方法使之更适合特殊的需要，这就是继承。继承是子类自动共享父类数据和方法的机制，这是类之间的一种关系，提高了软件的可重用性和可扩展性。

多态：

多态是指程序中定义的引用变量所指向的具体类型和通过该引用变量发出的方法调用在编程时并不确定，而是在程序运行期间才确定，即一个引用变量倒底会指向哪个类的实例对象，该引用变量发出的方法调用到底是哪个类中实现的方法，必须在由程序运行期间才能决定。因为在程序运行时才确定具体的类，这样，不用修改源程序代码，就可以让引用变量绑定到各种不同的类实现上，从而导致该引用调用的具体方法随之改变，即不修改程序代码就可以改变程序运行时所绑定的具体代码，让程序可以选择多个运行状态，这就是多态性。多态性增强了软件的灵活性和扩展性。例如，下面代码中的 UserDao 是一个接口，它定义引用变量 userDao 指向的实例对象由 daofactory.getDao()在执行的时候返回，有时候指向的是 UserJdbcDao 这个实现，有时候指向的是 UserHibernateDao 这个实现，这样，不用修改源代码，就可以改变 userDao 指向的具体类实现，从而导致 userDao.insertUser()方法调用的具体代码也随之改变，即有时候调用的是 UserJdbcDao 的 insertUser 方法，有时候调用的是 UserHibernateDao 的 insertUser 方法：

```
UserDao userDao =daofactory.getDao();
userDao.insertUser(user);
```

比喻：人吃饭，你看到的是左手，还是右手？

23、java 中实现多态的机制是什么？

靠的是父类或接口定义的引用变量可以指向子类或具体实现类的实例对象，而程序调用的方法在运行期才动态绑定，就是引用变量所指向的具体实例对象的方法，也就是内存里正在运行的那个对象的方法，而不是引用变量的类型中定义的方法。

24、abstract class 和 interface 有什么区别?

含有 abstract 修饰符的 class 即为抽象类，abstract 类不能创建的实例对象。含有 abstract 方法的类必须定义为abstract class，abstract class 类中的方法不必是抽象的。abstract class

类中定义抽象方法必须在具体(Concrete)子类中实现，所以，不能有抽象构造方法或抽象静态方法。如果的子类没有实现抽象父类中的所有抽象方法，那么子类也必须定义为 abstract 类型。

接口（interface）可以说成是抽象类的一种特例，接口中的所有方法都必须是抽象的。接口中的方法定义默认为 public abstract 类型，接口中的成员变量类型默认为 public static final。

下面比较一下两者的语法区别：

1.抽象类可以有构造方法，接口中不能有构造方法。

2.抽象类中可以有普通成员变量，接口中没有普通成员变量

3.抽象类中可以包含非抽象的普通方法，接口中的所有方法必须都是抽象的，不能有非抽象的普通方法。

4. 抽象类中的抽象方法的访问类型可以是 public，protected 和（默认类型,虽然

eclipse 下不报错，但应该也不行），但接口中的抽象方法只能是 public 类型的，并且默认即为 public abstract 类型。

5. 抽象类中可以包含静态方法，接口中不能包含静态方法

6. 抽象类和接口中都可以包含静态成员变量，抽象类中的静态成员变量的访问类型可以任意，但接口中定义的变量只能是 public static final 类型，并且默认即为 public static final 类型。

7. 一个类可以实现多个接口，但只能继承一个抽象类。

下面接着再说说两者在应用上的区别：

接口更多的是在系统架构设计方法发挥作用，主要用于定义模块之间的通信契约。而抽象类在代码实现方面发挥作用，可以实现代码的重用，例如，模板方法设计模式是抽象类的一个典型应用，假设某个项目的所有 Servlet 类都要用相同的方式进行权限判断、记录访问日志和处理异常，那么就可以定义一个抽象的基类，让所有的 Servlet 都继承这个抽象基类，在抽象基类的 service 方法中完成权限判断、记录访问日志和处理异常的代码，在各个子类中只是完成各自的业务逻辑代码，伪代码如下：

```
public abstract classBaseServlet extends HttpServlet{
        public final void service(HttpServletRequest request,HttpServletResponse
response) throws IOExcetion,ServletException        {
                记录访问日志
                进行权限判断
if(具有权限){
        try{
```

```
                    doService(request,response);
}
        catch(Excetpion e)   {
                        记录异常信息
        }
}
         }
         protected abstract void doService(HttpServletRequest
request,HttpServletResponse response) throws IOExcetion,ServletException;
//注意访问权限定义成 protected，显得既专业，又严谨，因为它是专门给子类用的
}

public class MyServlet1 extendsBaseServlet
{
protected voiddoService(HttpServletRequest request, HttpServletResponse response)
throwsIOExcetion,ServletException
         {
                本 Servlet 只处理的具体业务逻辑代码
         }

}
```

父类方法中间的某段代码不确定，留给子类干，就用模板方法设计模式。

备注：这道题的思路是先从总体解释抽象类和接口的基本概念，然后再比较两者的语法细节，最后再说两者的应用区别。比较两者语法细节区别的条理是：先从一个类中的构造方法、普通成员变量和方法（包括抽象方法），静态变量和方法，继承性等6个方面逐一去比较回答，接着从第三者继承的角度的回答，特别是最后用了一个典型的例子来展现自己深厚的技术功底。

25、abstract 的 method 是否可同时是 static,是否可同时是 native，是否可同时是 synchronized?

abstract 的 method 不可以是 static 的，因为抽象的方法是要被子类实现的，而 static 与子类扯不上关系！

native 方法表示该方法要用另外一种依赖平台的编程语言实现的，不存在着被子类实现的问题，所以，它也不能是抽象的，不能与 abstract 混用。例如，FileOutputSteam 类要硬件打交道，底层的实现用的是操作系统相关的 api 实现，例如，在 windows 用 c 语言实现的，所以，查看 jdk 的源代码，可以发现 FileOutputStream 的 open 方法的定义如下：

private native void open(Stringname) throws FileNotFoundException;

如果我们要用 java 调用别人写的 c 语言函数，我们是无法直接调用的，我们需要按照 java 的要求写一个 c 语言的函数，又我们的这个 c 语言函数去调用别人的 c 语言函数。由于我们的 c 语言函数是按 java 的要求来写的，我们这个 c 语言函数就可以与 java 对接上，java 那边的对接方式就是定义出与我们这个 c 函数相对应的方法，java 中对应的方法不需要写具体的代码，但需要在前面声明 native。

关于 synchronized 与 abstract 合用的问题，我觉得也不行，因为在我几年的学习和开发中，从来没见到过这种情况，并且我觉得 synchronized 应该是作用在一个具体的方法上才有意义。而且，方法上的 synchronized 同步所使用的同步锁对象是 this，而抽象方法上无法确定 this 是什么。

26、什么是内部类？Static Nested Class 和 Inner Class 的不同。

内部类就是在一个类的内部定义的类，内部类中不能定义静态成员（静态成员不是对象的特性，只是为了找一个容身之处，所以需要放到一个类中而已，这么一点小事，你还要把它放到类内部的一个类中，过分了啊！提供内部类，不是为让你干这种事情，无聊，不让你干。我想可能是既然静态成员类似 c 语言的全局变量，而内部类通常是用于创建内部对象用的，所以，把“全局变量”放在内部类中就是毫无意义的事情，既然是毫无意义的事情，就应该被禁止），内部类可以直接访问外部类中的成员变量，内部类可以定义在外部类的方法外面，也可以定义在外部类的方法体中，如下所示：

```
public class Outer
{
        int out_x   = 0;
        public void method()
            {
             Inner1 inner1 = new Inner1();
             public class Inner2   //在方法体内部定义的内部类
             {
                  public method()
                  {
                        out_x = 3;
```

```
                }
            }
            Inner2 inner2 = new Inner2();
        }

        public class Inner1   //在方法体外面定义的内部类
        {
        }

}
```

在方法体外面定义的内部类的访问类型可以是 public,protecte,默认的，private 等4种类型，这就好像类中定义的成员变量有4种访问类型一样，它们决定这个内部类的定义对其他类是否可见；对于这种情况，我们也可以在外面创建内部类的实例对象，创建内部类的实例对象时，一定要先创建外部类的实例对象，然后用这个外部类的实例对象去创建内部类的实例对象，代码如下：

```
Outer outer = new Outer();
Outer.Inner1 inner1 = outer.new Innner1();
```

在方法内部定义的内部类前面不能有访问类型修饰符，就好像方法中定义的局部变量一样，但这种内部类的前面可以使用 final 或 abstract 修饰符。这种内部类对其他类是不可见的其他类无法引用这种内部类，但是这种内部类创建的实例对象可以传递给其他类访问。这种内部类必须是先定义，后使用，即内部类的定义代码必须出现在使用该类之前，这与方法中的局部变量必须先定义后使用的道理也是一样的。这种内部类可以访问方法体中的局部变量，但是，该局部变量前必须加 final 修饰符。

对于这些细节，只要在 eclipse 写代码试试，根据开发工具提示的各类错误信息就可以马上了解到。

在方法体内部还可以采用如下语法来创建一种匿名内部类，即定义某一接口或类的子类的同时，还创建了该子类的实例对象，无需为该子类定义名称：

```
public class Outer
{
        public void start()
```

```
{
    new Thread(
new Runable(){
        public void run(){};
}
).start();
    }
}
```

最后，在方法外部定义的内部类前面可以加上 static 关键字，从而成为 Static Nested Class，它不再具有内部类的特性，所有，从狭义上讲，它不是内部类。Static Nested Class 与普通类在运行时的行为和功能上没有什么区别，只是在编程引用时的语法上有一些差别，它可以定义成 public、protected、默认的、private 等多种类型，而普通类只能定义成 public 和默认的这两种类型。在外面引用 Static Nested Class 类的名称为"外部类名.内部类名"。在外面不需要创建外部类的实例对象，就可以直接创建 Static Nested Class，例如，假设 Inner 是定义在 Outer 类中的 Static Nested Class，那么可以使用如下语句创建 Inner 类：

Outer.Inner inner = newOuter.Inner();

由于 static Nested Class 不依赖于外部类的实例对象，所以，**static Nested Class** 能访问外部类的非 **static** 成员变量**(**不能直接访问，需要创建外部类实例才能访问非静态变量**)**。当在外部类中访问 Static Nested Class 时，可以直接使用 Static Nested Class 的名字，而不需要加上外部类的名字了，在 Static Nested Class 中也可以直接引用外部类的 static 的成员变量，不需要加上外部类的名字。

在静态方法中定义的内部类也是 Static Nested Class，这时候不能在类前面加 static 关键字，静态方法中的 Static Nested Class 与普通方法中的内部类的应用方式很相似，它除了可以直接访问外部类中的 static 的成员变量，还可以访问静态方法中的局部变量，但是，该局部变量前必须加 final 修饰符。

备注：首先根据你的印象说出你对内部类的总体方面的特点：例如，在两个地方可以定义，可以访问外部类的成员变量，不能定义静态成员，这是大的特点。然后再说一些细节方面的知识，例如，几种定义方式的语法区别，静态内部类，以及匿名内部类。

27、内部类可以引用它的包含类的成员吗？有没有什么限制？

完全可以。如果不是静态内部类，那没有什么限制！

如果你把静态嵌套类当作内部类的一种特例,那在这种情况下不可以访问外部类的普通成员变量，而只能访问外部类中的静态成员，例如，下面的代码：

```
class Outer
{
static int x;
static class Inner
{
        void test()
        {
                syso(x);
        }
}
}
```

答题时，也要能察言观色，揣摩提问者的心思，显然人家希望你说的是静态内部类不能访问外部类的成员，但你一上来就顶牛，这不好，要先顺着人家，让人家满意，然后再说特殊情况，让人家吃惊。

28、Anonymous Inner Class (匿名内部类)是否可以 extends(继承)其它类，是否可以 implements(实现)interface(接口)?

可以继承其他类或实现其他接口。不仅是可以，而是必须!

29、super.getClass()方法调用

下面程序的输出结果是多少？

```
importjava.util.Date;
public class Test extends Date{
    public static void main(String[] args) {
        new Test().test();
    }

    public void test(){
```

```
        System.out.println(super.getClass().getName());
    }
}
```

很奇怪，结果是 Test

这属于脑筋急转弯的题目，在一个 qq 群有个网友正好问过这个问题，我觉得挺有趣，就研究了一下，没想到今天还被你面到了，哈哈。

在 test 方法中，直接调用 getClass().getName()方法，返回的是 Test 类名

由于 getClass()在 Object 类中定义成了 final，子类不能覆盖该方法，所以，在

test 方法中调用 getClass().getName()方法，其实就是在调用从父类继承的 getClass()方法，等效于调用 super.getClass().getName()方法，所以，super.getClass().getName()方法返回的也应该是 Test。

如果想得到父类的名称，应该用如下代码：

getClass().getSuperClass().getName();

30、String 是最基本的数据类型吗?

基本数据类型包括 byte、int、char、long、float、double、boolean 和 short。

java.lang.String 类是 final 类型的，因此不可以继承这个类、不能修改这个类。为了提高效率节省空间，我们应该用 StringBuffer 类

31、String s = "Hello";s = s + " world!";这两行代码执行后，原始的 String 对象中的内容到底变了没有？

没有。因为 String 被设计成不可变(immutable)类，所以它的所有对象都是不可变对象。在这段代码中，s 原先指向一个 String 对象，内容是 "Hello"，然后我们对 s 进行了+操作，那么 s 所指向的那个对象是否发生了改变呢？答案是没有。这时，s 不指向原来那个对象了，而指向了另一个 String 对象，内容为"Hello world!"，原来那个对象还存在于内存之中，只是 s 这个引用变量不再指向它了。

通过上面的说明，我们很容易导出另一个结论，如果经常对字符串进行各种各样的修改，或者说，不可预见的修改，那么使用 String 来代表字符串的话会引起很大的内存开销。因为 String 对象建立之后不能再改变，所以对于每一个不同的字符串，都需要一个 String 对象来表示。这时，应该考虑使用 StringBuffer 类，它允许修改，而不是每个不同的字符串都要生成一个新的对象。并且，这两种类的对象转换十分容易。

同时，我们还可以知道，如果要使用内容相同的字符串，不必每次都 new 一个 String。例如我们要在构造器中对一个名叫 s 的 String 引用变量进行初始化，把它设置为初始值，应

当这样做：

```
public class Demo {
private String s;
...
public Demo {
s = "Initial Value";
}
...
}
```

而非

```
s = new String("Initial Value");
```

后者每次都会调用构造器，生成新对象，性能低下且内存开销大，并且没有意义，因为 String 对象不可改变，所以对于内容相同的字符串，只要一个 String 对象来表示就可以了。也就说，多次调用上面的构造器创建多个对象，他们的 String 类型属性 s 都指向同一个对象。

上面的结论还基于这样一个事实：对于字符串常量，如果内容相同，Java 认为它们代表同一个 String 对象。而用关键字 new 调用构造器，总是会创建一个新的对象，无论内容是否相同。

至于为什么要把 String 类设计成不可变类，是它的用途决定的。其实不只 String，很多 Java 标准类库中的类都是不可变的。在开发一个系统的时候，我们有时候也需要设计不可变类，来传递一组相关的值，这也是面向对象思想的体现。不可变类有一些优点，比如因为它的对象是只读的，所以多线程并发访问也不会有任何问题。当然也有一些缺点，比如每个不同的状态都要一个对象来代表，可能会造成性能上的问题。所以 Java 标准类库还提供了一个可变版本，即 StringBuffer。

32、是否可以继承 String 类?

String 类是 final 类故不可以继承。

33、String s = new String("xyz");创建了几个 String Object?二者之间有什么区别？

两个或一个，”xyz”对应一个对象，这个对象放在字符串常量缓冲区，常量”xyz”不管出现多少遍，都是缓冲区中的那一个。New String 每写一遍，就创建一个新的对象，它一句那个常量”xyz”对象的内容来创建出一个新 String 对象。如果以前就用过’xyz’，这句代表就不会创建”xyz”自己了，直接从缓冲区拿。

34、String 和 StringBuffer 的区别

JAVA 平台提供了两个类：String 和 StringBuffer，它们可以储存和操作字符串，即包含多个字符的字符数据。这个 String 类提供了数值不可改变的字符串。而这个 StringBuffer 类提供的字符串进行修改。当你知道字符数据要改变的时候你就可以使用 StringBuffer。典型地，你可以使用 StringBuffers 来动态构造字符数据。另外，String 实现了 equals 方法，new String(“abc”).equals(newString(“abc”)的结果为 true,而 StringBuffer 没有实现 equals 方法，所以，new StringBuffer(“abc”).equals(newStringBuffer(“abc”)的结果为 false。

接着要举一个具体的例子来说明，我们要把1到100的所有数字拼起来，组成一个串。

```
StringBuffer sbf = new StringBuffer();
for(int i=0;i<100;i++)
{
        sbf.append(i);
}
```

上面的代码效率很高，因为只创建了一个 StringBuffer 对象，而下面的代码效率很低，因为创建了101个对象。

```
String str = new String();
for(int i=0;i<100;i++)
{
        str = str + i;
}
```

在讲两者区别时，应把循环的次数搞成10000，然后用 endTime-beginTime 来比较两者执行的时间差异，最后还要讲讲 StringBuilder 与 StringBuffer 的区别。

String 覆盖了 equals 方法和 hashCode 方法，而 StringBuffer 没有覆盖 equals 方法和 hashCode 方法，所以，将 **StringBuffer** 对象存储进 **Java** 集合类中时会出现问题。

35、如何把一段逗号分割的字符串转换成一个数组?

如果不查 jdk api，我很难写出来！我可以说说我的思路：

1 用正则表达式，代码大概为：String [] result = orgStr.split(“,”);

2 用 StingTokenizer ,代码为：StringTokenizer tokener = StringTokenizer(orgStr,”,”);

String [] result =new String[tokener .countTokens()];

Int i=0;

while(tokener.hasNext(){result[i++]=toker.nextToken();}

36、数组有没有 length()这个方法? String 有没有 length()这个方法？

数组没有 length()这个方法，有 length 的属性。String 有有 length()这个方法。

37、下面这条语句一共创建了多少个对象：String s="a"+"b"+"c"+"d";

答：对于如下代码：

```
String s1 = "a";

String s2 = s1 + "b";

String s3 = "a" + "b";

System.out.println(s2 == "ab");

System.out.println(s3 == "ab");
```

第一条语句打印的结果为 false，第二条语句打印的结果为 true，这说明 javac 编译可以对字符串常量直接相加的表达式进行优化，不必要等到运行期去进行加法运算处理，而是在编译时去掉其中的加号，直接将其编译成一个这些常量相连的结果。

题目中的第一行代码被编译器在编译时优化后，相当于直接定义了一个"abcd"的字符串，所以，上面的代码应该只创建了一个 String 对象。写如下两行代码，

```
String s ="a" + "b" + "c" + "d";

System.out.println(s== "abcd");
```

最终打印的结果应该为 true。

38、try {}里有一个 return 语句，那么紧跟在这个 try 后的 finally {}里的 code 会不会被执行，什么时候被执行，在 return 前还是后?

也许你的答案是在 return 之前，但往更细地说，我的答案是在 return 中间执行，请看下面程序代码的运行结果：

```
public classTest {


    /**

     * @paramargs add by zxx ,Dec 9, 2008

     */

    public static void main(String[] args) {
```

```
        // TODO Auto-generated method stub
        System.out. println (new Test().test());;
    }

    static int test()
    {
        int x = 1;
        try
        {
            Return x;
        }
        finally
        {
            ++x;
        }
    }

}
```

---------执行结果 ---------

1

运行结果是1，为什么呢？主函数调用子函数并得到结果的过程，好比主函数准备一个空罐子，当子函数要返回结果时，先把结果放在罐子里，然后再将程序逻辑返回到主函数。所谓返回，就是子函数说，我不运行了，你主函数继续运行吧，这没什么结果可言，结果是在说这话之前放进罐子里的。

39、下面的程序代码输出的结果是多少？

```
public class   smallT
{
        public static void   main(String args[])
        {
```

```
        smallT  t  = new smallT();
        int  b =  t.get();
        System.out.println(b);
    }

    public int  get()
    {
        try
        {
            Return  1 ;
        }
        finally
        {
            Return  2 ;
        }
    }
}
```

返回的结果是2。

我可以通过下面一个例子程序来帮助我解释这个答案，从下面例子的运行结果中可以发现，try 中的 return 语句调用的函数先于 finally 中调用的函数执行，也就是说 return 语句先执行，finally 语句后执行，所以，返回的结果是2。Return 并不是让函数马上返回，而是 return 语句执行后，将把返回结果放置进函数栈中，此时函数并不是马上返回，它要执行 finally 语句后才真正开始返回。

在讲解答案时可以用下面的程序来帮助分析：

```
public classTest {

    /**
     * @paramargs add by zxx ,Dec 9, 2008
     */
    public static voidmain(String[] args) {
```

```
        // TODO Auto-generated method stub
        System.out.println(newTest().test());;
    }

    int test()
    {
        try
        {
            return func1();
        }
        finally
        {
            return func2();
        }
    }

    int func1()
    {
         System.out.println("func1");
        return 1;
    }
    int func2()
    {
        System.out.println("func2");
        return 2;
    }
}
```

-----------执行结果-----------------

func1

func2

2

结论：finally 中的代码比 return 和 break 语句后执行

40、final, finally, finalize 的区别。

final 用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。

内部类要访问局部变量，局部变量必须定义成 final 类型，例如，一段代码……

finally 是异常处理语句结构的一部分，表示总是执行。

finalize 是 Object 类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。JVM 不保证此方法总被调用

41、运行时异常与一般异常有何异同？

异常表示程序运行过程中可能出现的非正常状态，运行时异常表示虚拟机的通常操作中可能遇到的异常，是一种常见运行错误。java 编译器要求方法必须声明抛出可能发生的非运行时异常，但是并不要求必须声明抛出未被捕获的运行时异常。

42、error 和 exception 有什么区别?

error 表示恢复不是不可能但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况。 exception 表示一种设计或实现问题。也就是说，它表示如果程序运行正常，从不会发生的情况。

43、Java 中的异常处理机制的简单原理和应用。

异常是指 java 程序运行时（非编译）所发生的非正常情况或错误，与现实生活中的事件很相似，现实生活中的事件可以包含事件发生的时间、地点、人物、情节等信息，可以用一个对象来表示，Java 使用面向对象的方式来处理异常，它把程序中发生的每个异常也都分别封装到一个对象来表示的，该对象中包含有异常的信息。

Java 对异常进行了分类，不同类型的异常分别用不同的 Java 类表示，所有异常的根类为 java.lang.Throwable，Throwable 下面又派生了两个子类：Error 和 Exception，Error 表示应用程序本身无法克服和恢复的一种严重问题，程序只有死的份了，例如，说内存溢出和线程死锁等系统问题。Exception 表示程序还能够克服和恢复的问题，其中又分为系统异常和普通异常，系统异常是软件本身缺陷所导致的问题，也就是软件开发人员考虑不周所导致的

问题，软件使用者无法克服和恢复这种问题，但在这种问题下还可以让软件系统继续运行或者让软件死掉，例如，数组脚本越界（ArrayIndexOutOfBoundsException），空指针异常（NullPointerException）、类转换异常（ClassCastException）；普通异常是运行环境的变化或异常所导致的问题，是用户能够克服的问题，例如，网络断线，硬盘空间不够，发生这样的异常后，程序不应该死掉。

java 为系统异常和普通异常提供了不同的解决方案，编译器强制普通异常必须 try..catch 处理或用 throws 声明继续抛给上层调用方法处理，所以普通异常也称为 checked 异常，而系统异常可以处理也可以不处理，所以，编译器不强制用 try..catch 处理或用 throws 声明，所以系统异常也称为 unchecked 异常。

提示答题者：就按照三个级别去思考：虚拟机必须宕机的错误，程序可以死掉也可以不死掉的错误，程序不应该死掉的错误；

44、请写出你最常见到的 5 个 runtime exception。

这道题主要考你的代码量到底多大，如果你长期写代码的，应该经常都看到过一些系统方面的异常，你不一定真要回答出5个具体的系统异常，但你要能够说出什么是系统异常，以及几个系统异常就可以了，当然，这些异常完全用其英文名称来写是最好的，如果实在写不出，那就用中文吧，有总比没有强！

所谓系统异常，就是…..，它们都是 RuntimeException 的子类，在 jdk doc 中查 RuntimeException 类，就可以看到其所有的子类列表，也就是看到了所有的系统异常。我比较有印象的系统异常有：NullPointerException、ArrayIndexOutOfBoundsException、ClassCastException。

45、JAVA 语言如何进行异常处理，关键字：throws,throw,try,catch,finally 分别代表什么意义？在 try 块中可以抛出异常吗？

throws 捕获并向外抛出异常
throw 抛出异常
try catch 是内部捕获异常并做自定义处理
finally 是无论是否有异常都会被处理的语句，除非在 finally 前存在被执行的 System.exit(int i)时除外

46、java 中有几种方法可以实现一个线程？用什么关键字修饰同步方法? stop() 和 suspend()方法为何不推荐使用？

java5以前，有如下两种：

第一种：

new Thread(){}.start();这表示调用 Thread 子类对象的 run 方法，new Thread(){}表示一个 Thread 的匿名子类的实例对象，子类加上 run 方法后的代码如下：

```
new Thread(){
    public void run(){
    }
}.start();
```

第二种：

new Thread(new Runnable(){}).start();这表示调用 Thread 对象接受的 Runnable 对象的 run 方法，new Runnable(){}表示一个 Runnable 的匿名子类的实例对象,runnable 的子类加上 run 方法后的代码如下：

```
new Thread(new Runnable(){
                public voidrun(){
                }
        }
    ).start();
```

从 java5开始，还有如下一些线程池创建多线程的方式：

```
ExecutorService pool = Executors.newFixedThreadPool(3)
for(int i=0;i<10;i++)
{
 pool.execute(newRunable(){public void run(){}});
}
Executors.newCachedThreadPool().execute(new Runable(){publicvoid run(){}});
Executors.newSingleThreadExecutor().execute(new Runable(){publicvoid run(){}});
```

有两种实现方法，分别使用 new Thread()和 new Thread(runnable)形式，第一种直接调用 thread 的 run 方法，所以，我们往往使用 Thread 子类，即 new SubThread()。第二种调用

runnable 的 run 方法。

有两种实现方法，分别是继承 Thread 类与实现 Runnable 接口

用 synchronized 关键字修饰同步方法

反对使用 stop()，是因为它不安全。它会解除由线程获取的所有锁定，而且如果对象处于一种不连贯状态，那么其他线程能在那种状态下检查和修改它们。结果很难检查出真正的问题所在。suspend()方法容易发生死锁。调用 suspend()的时候，目标线程会停下来，但却仍然持有在这之前获得的锁定。此时，其他任何线程都不能访问锁定的资源，除非被"挂起"的线程恢复运行。对任何线程来说，如果它们想恢复目标线程，同时又试图使用任何一个锁定的资源，就会造成死锁。所以不应该使用 suspend()，而应在自己的 Thread 类中置入一个标志，指出线程应该活动还是挂起。若标志指出线程应该挂起，便用 wait()命其进入等待状态。若标志指出线程应当恢复，则用一个 notify()重新启动线程。

47、sleep()和 wait()有什么区别?

（网上的答案：sleep 是线程类（Thread）的方法，导致此线程暂停执行指定时间，给执行机会给其他线程，但是监控状态依然保持，到时后会自动恢复。调用 sleep 不会释放对象锁。 wait 是 Object 类的方法，对此对象调用 wait 方法导致本线程放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象发出 notify 方法（或 notifyAll）后本线程才进入对象锁定池准备获得对象锁进入运行状态。）

sleep 就是正在执行的线程主动让出 cpu，cpu 去执行其他线程，在 sleep 指定的时间过后，cpu 才会回到这个线程上继续往下执行，如果当前线程进入了同步锁，sleep 方法并不会释放锁，即使当前线程使用 sleep 方法让出了 cpu，但其他被同步锁挡住了的线程也无法得到执行。wait 是指在一个已经进入了同步锁的线程内，让自己暂时让出同步锁，以便其他正在等待此锁的线程可以得到同步锁并运行，只有其他线程调用了 notify 方法（notify 并不释放锁，只是告诉调用过 wait 方法的线程可以去参与获得锁的竞争了，但不是马上得到锁，因为锁还在别人手里，别人还没释放。如果 notify 方法后面的代码还有很多，需要这些代码执行完后才会释放锁，可以在 notfiy 方法后增加一个等待和一些代码，看看效果），调用 wait 方法的线程就会解除 wait 状态和程序可以再次得到锁后继续向下运行。对于 wait 的讲解一定要配合例子代码来说明，才显得自己真明白。

```
package com.huawei.interview;


publicclass MultiThread {
```

```
/**
 * @paramargs
 */
public static voidmain(String[] args) {
    // TODO Auto-generated method stub
    new Thread(newThread1()).start();
    try {
        Thread.sleep(10);
    } catch (InterruptedException e) {
        // TODO Auto-generated catchblock
        e.printStackTrace();
    }
    new Thread(newThread2()).start();
}


private static classThread1implements Runnable
{

    @Override
    public void run() {
        // TODO Auto-generated methodstub
//由于这里的 Thread1和下面的 Thread2内部 run 方法要用同一对象作为监视器，我们这里不能用 this，因为在 Thread2里面的 this 和这个 Thread1的 this 不是同一个对象。我们用 MultiThread.class 这个字节码对象，当前虚拟机里引用这个变量时，指向的都是同一个对象。
        synchronized (MultiThread.class){

            System.out.println("enterthread1...");

            System.out.println("thread1is waiting");
```

```
                try {
            //释放锁有两种方式，第一种方式是程序自然离开监视器的范围，也就是离开了 synchronized 关键字管辖的代码范围，另一种方式就是在 synchronized 关键字管辖的代码内部调用监视器对象的 wait 方法。这里，使用 wait 方法释放锁。
                    MultiThread.class.wait();
                } catch(InterruptedException e) {
                    // TODO Auto-generatedcatch block
                    e.printStackTrace();
                }

                System.out.println("thread1is going on...");
                System.out.println("thread1is being over!");
            }
        }

    }

    private static classThread2implements Runnable
    {

        @Override
        public void run() {
            // TODO Auto-generated methodstub
            synchronized (MultiThread.class){

                System.out.println("enterthread2...");

                System.out.println("thread2notify other thread can release wait status..");
//由于 notify 方法并不释放锁，即使 thread2调用下面的 sleep 方法休息了10毫秒，但 thread1仍然不会执行，因为 thread2没有释放锁，所以 Thread1无法得不到锁。
```

```
                MultiThread.class.notify();

                System.out.println("thread2is sleeping ten millisecond...");
                try {
                    Thread.sleep(10);
                } catch (InterruptedExceptione) {
                    // TODO Auto-generatedcatch block
                    e.printStackTrace();
                }

                System.out.println("thread2is going on...");
                System.out.println("thread2is being over!");

            }
        }

    }

}
```

48、同步和异步有何异同，在什么情况下分别使用他们？举例说明。

如果数据将在线程间共享。例如正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就是共享数据，必须进行同步存取。

当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程，在很多情况下采用异步途径往往更有效率。

49. 下面两个方法同步吗？（自己发明）

```
    class Test
```

```
{
synchronizedstatic   void say Hello3()
        {


        }


        synchronizedvoid   getX(){}
}
```

50、多线程有几种实现方法?同步有几种实现方法?

多线程有两种实现方法，分别是继承 Thread 类与实现 Runnable 接口

同步的实现方面有两种，分别是 synchronized,wait 与 notify

wait():使一个线程处于等待状态，并且释放所持有的对象的 lock。

sleep():使一个正在运行的线程处于睡眠状态，是一个静态方法，调用此方法要捕捉 InterruptedException(中断异常)异常。

notify():唤醒一个处于等待状态的线程，注意的是在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由 JVM 确定唤醒哪个线程，而且不是按优先级。

Allnotity():唤醒所有处入等待状态的线程，注意并不是给所有唤醒线程一个对象的锁，而是让它们竞争。

51、启动一个线程是用 run()还是 start()? .

启动一个线程是调用 start()方法，使线程就绪状态，以后可以被调度为运行状态，一个线程必须关联一些具体的执行代码，run()方法是该线程所关联的执行代码。

52、当一个线程进入一个对象的一个 synchronized 方法后，其它线程是否可进入此对象的其它方法?

分几种情况：

1. 其他方法前是否加了 synchronized 关键字，如果没加，则能。
2. 如果这个方法内部调用了 wait，则可以进入其他 synchronized 方法。
3. 如果其他个方法都加了 synchronized 关键字，并且内部没有调用 wait，则不能。

4. 如果其他方法是 static，它用的同步锁是当前类的字节码，与非静态的方法不能同步，因为非静态的方法用的是 this。

53、线程的基本概念、线程的基本状态以及状态之间的关系

一个程序中可以有多条执行线索同时执行，一个线程就是程序中的一条执行线索，每个线程上都关联有要执行的代码，即可以有多段程序代码同时运行，每个程序至少都有一个线程，即 main 方法执行的那个线程。如果只是一个 cpu，它怎么能够同时执行多段程序呢？这是从宏观上来看的，cpu 一会执行 a 线索，一会执行 b 线索，切换时间很快，给人的感觉是 a,b 在同时执行，好比大家在同一个办公室上网，只有一条链接到外部网线，其实，这条网线一会为 a 传数据，一会为 b 传数据，由于切换时间很短暂，所以，大家感觉都在同时上网。

状态：就绪，运行，synchronize 阻塞，wait 和 sleep 挂起，结束。wait 必须在 synchronized 内部调用。

调用线程的 start 方法后线程进入就绪状态，线程调度系统将就绪状态的线程转为运行状态，遇到 synchronized 语句时，由运行状态转为阻塞，当 synchronized 获得锁后，由阻塞转为运行，在这种情况可以调用 wait 方法转为挂起状态，当线程关联的代码执行完后，线程变为结束状态。

54、简述 synchronized 和 java.util.concurrent.locks.Lock 的异同？

主要相同点：Lock 能完成 synchronized 所实现的所有功能

主要不同点：Lock 有比 synchronized 更精确的线程语义和更好的性能。synchronized 会自动释放锁，而 Lock 一定要求程序员手工释放，并且必须在 finally 从句中释放。Lock 还有更强大的功能，例如，它的 tryLock 方法可以非阻塞方式去拿锁。

举例说明（对下面的题用 lock 进行了改写）：

```
package com.huawei.interview;

import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

publicclass ThreadTest {
```

```
/**
 * @paramargs
 */

private int j;
private Lock lock =newReentrantLock();
public static voidmain(String[] args) {
    // TODO Auto-generated method stub
    ThreadTest tt = new ThreadTest();
    for(int i=0;i<2;i++)
    {
        new Thread(tt.new Adder()).start();
        new Thread(tt.new Subtractor()).start();
    }
}

private class SubtractorimplementsRunnable
{

    @Override
    public void run() {
        // TODO Auto-generated methodstub
        while(true)
        {
            /*synchronized (ThreadTest.this) {
                System.out.println("j--="+ j--);
                //这里抛异常了，锁能释放吗？
            }*/
            lock.lock();
            try
```

```
                {
                    System.out.println("j--="+ j--);
                }finally
                {
                    lock.unlock();
                }
            }
        }

    }

    private class Adderimplements Runnable
    {

        @Override
        public void run() {
            // TODO Auto-generated methodstub
            while(true)
            {
                /*synchronized (ThreadTest.this) {
                System.out.println("j++="+ j++);
                }*/
                lock.lock();
                 try
                {
                    System.out.println("j++="+ j++);
                }finally
                {
                    lock.unlock();
                }
            }
```

```
        }

    }
}
```

55、设计 **4** 个线程，其中两个线程每次对 **j** 增加 **1**，另外两个线程对 **j** 每次减少 **1**。写出程序。

以下程序使用内部类实现线程，对 j 增减的时候没有考虑顺序问题。

```
public class ThreadTest1
{
private int j;
public static void main(String args[]){
   ThreadTest1 tt=newThreadTest1();
   Inc inc=tt.new Inc();
   Dec dec=tt.new Dec();
   for(inti=0;i<2;i++){
        Thread t=newThread(inc);
        t.start();
            t=new Thread(dec);
        t.start();
        }
   }
private synchronized void inc(){
   j++;
  System.out.println(Thread.currentThread().getName()+"-inc:"+j);
   }
private synchronized void dec(){
   j--;
  System.out.println(Thread.currentThread().getName()+"-dec:"+j);
   }
class Inc implements Runnable{
```

```
    public void run(){
        for(inti=0;i<100;i++){
        inc();
        }
    }
}
class Dec implements Runnable{
    public void run(){
        for(inti=0;i<100;i++){
        dec();
        }
    }
}
}
```

----------随手再写的一个-------------

```
class A
{
JManger j =new JManager();
main()
{
    new A().call();
}

void call
{
    for(int i=0;i<2;i++)
    {
        new Thread(
            newRunnable(){ public void run(){while(true){j.accumulate()}}}
        ).start();
```

```
            new Thread(newRunnable(){ public void run(){while(true){j.sub()}}}).start();
        }
    }
}

class JManager
{
    private j = 0;

    public synchronized voidsubtract()
    {
            j--
    }

    public synchronized voidaccumulate()
    {
            j++;
    }

}
```

56、子线程循环 **10** 次，接着主线程循环 **100**，接着又回到子线程循环 **10** 次，接着再回到主线程又循环 **100**，如此循环 **50** 次，请写出程序。

最终的程序代码如下：

```
public class ThreadTest {

     /**
      * @paramargs
      */
```

```java
public static voidmain(String[] args) {
    // TODO Auto-generated method stub
    new ThreadTest().init();

}

public void init()
{
    final Business business =newBusiness();
    new Thread(
            new Runnable()
            {

                public voidrun() {
                    for(inti=0;i<50;i++)
                    {
                        business.SubThread(i);
                    }
                }

            }

    ).start();

    for(int i=0;i<50;i++)
    {
        business.MainThread(i);
    }
}

private class Business
```

```
{
    booleanbShouldSub =true;//这里相当于定义了控制该谁执行的一个信号灯
    public synchronized voidMainThread(int i)
    {
        if(bShouldSub)
            try {
                this.wait();
            } catch(InterruptedException e) {
                // TODO Auto-generatedcatch block
                e.printStackTrace();
            }

        for(int j=0;j<5;j++)
        {
            System.out.println(Thread.currentThread().getName()+ ":i=" + i +",j=" + j);
        }
        bShouldSub =true;
        this.notify();

    }



    public synchronized voidSubThread(int i)
    {
        if(!bShouldSub)
            try {
                this.wait();
            } catch (InterruptedExceptione) {
                // TODO Auto-generatedcatch block
                e.printStackTrace();
            }
```

```java
                for(intj=0;j<10;j++)
                {
                    System.out.println(Thread.currentThread().getName()+ ":i=" + i +",j=" + j);
                }
                bShouldSub =false;
                this.notify();
            }
        }
}
```

备注：不可能一上来就写出上面的完整代码，最初写出来的代码如下，问题在于两个线程的代码要参照同一个变量，即这两个线程的代码要共享数据，所以，把这两个线程的执行代码搬到同一个类中去：

```java
package com.huawei.interview.lym;

publicclass ThreadTest {

    private static booleanbShouldMain=false;

    public static void main(String[]args) {
        // TODO Auto-generated method stub
        /*new Thread(){
        public void run()
        {
            for(int i=0;i<50;i++)
            {
                for(int j=0;j<10;j++)
                {
                    System.out.println("i="+ i + ",j=" + j);
```

```
            }
        }
    }
}.start();*/

    //final String str = newString("");
    new Thread(
            new Runnable()
            {
                public voidrun()
                {
                    for(inti=0;i<50;i++)
                    {
                        synchronized(ThreadTest.class) {
                            if(bShouldMain)
                            {
                                try {
                                    ThreadTest.class.wait();}
                                catch(InterruptedException e) {
                                    e.printStackTrace();
                                }
                            }
                            for(intj=0;j<10;j++)
                            {
                                System.out.println(
                                        Thread.currentThread().getName()+
                                        "i="+ i + ",j=" + j);
                            }
```

```
                    bShouldMain= true;
                    ThreadTest.class.notify();
                }
            }
        }
    }
).start();

for(int i=0;i<50;i++)
{
    synchronized (ThreadTest.class){
        if(!bShouldMain)
        {
            try {
                ThreadTest.class.wait();}
            catch(InterruptedException e) {
                e.printStackTrace();
            }
        }
        for(intj=0;j<5;j++)
        {
            System.out.println(
                    Thread.currentThread().getName()+
                    "i=" + i +",j=" + j);
        }
        bShouldMain =false;
        ThreadTest.class.notify();
    }
}
}
```

}

下面使用 jdk5中的并发库来实现的：

```
import java.util.concurrent.Executors;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import java.util.concurrent.locks.Condition;

public class ThreadTest
{
    private static Locklock = new ReentrantLock();
    private staticCondition subThreadCondition = lock.newCondition();
    private staticboolean bBhouldSubThread = false;
    public static voidmain(String [] args)
    {
        ExecutorServicethreadPool = Executors.newFixedThreadPool(3);
        threadPool.execute(newRunnable(){
            publicvoid run()
            {
                for(inti=0;i<50;i++)
                {
                    lock.lock();
                    try
                    {
                        if(!bBhouldSubThread)
                            subThreadCondition.await();
                        for(intj=0;j<10;j++)
                        {

System.out.println(Thread.currentThread().getName()+ ",j=" + j);
                        }
```

```
                                        bBhouldSubThread= false;
                                        subThreadCondition.signal();
                                }catch(Exceptione)
                                {
                                }
                                finally
                                {
                                        lock.unlock();
                                }
                        }
                }

        });
        threadPool.shutdown();
        for(inti=0;i<50;i++)
        {
                        lock.lock();
                        try
                        {
                                if(bBhouldSubThread)
                                                subThreadCondition.await();
                                for(intj=0;j<10;j++)
                                {

System.out.println(Thread.currentThread().getName()+ ",j=" + j);
                                }
                                bBhouldSubThread= true;
                                subThreadCondition.signal();
                        }catch(Exceptione)
                        {
                        }
```

```
                    finally
                    {
                            lock.unlock();
                    }
            }
        }
}
```

57、介绍 Collection 框架的结构

答：随意发挥题，天南海北谁便谈，只要让别觉得你知识渊博，理解透彻即可。

58、Collection 框架中实现比较要实现什么接口

comparable/comparator

59、ArrayList 和 Vector 的区别

答：

这两个类都实现了 List 接口（List 接口继承了 Collection 接口），他们都是有序集合，即存储在这两个集合中的元素的位置都是有顺序的，相当于一种动态的数组，我们以后可以按位置索引号取出某个元素，，并且其中的数据是允许重复的，这是 HashSet 之类的集合的最大不同处，HashSet 之类的集合不可以按索引号去检索其中的元素，也不允许有重复的元素（本来题目问的与 hashset 没有任何关系，但为了说清楚 ArrayList 与 Vector 的功能，我们使用对比方式，更有利于说明问题）。

接着才说 ArrayList 与 Vector 的区别，这主要包括两个方面：.

（1）同步性：

Vector 是线程安全的，也就是说是它的方法之间是线程同步的，而 ArrayList 是线程序不安全的，它的方法之间是线程不同步的。如果只有一个线程会访问到集合，那最好是使用 ArrayList，因为它不考虑线程安全，效率会高些；如果有多个线程会访问到集合，那最好是使用 Vector，因为不需要我们自己再去考虑和编写线程安全的代码。

备注：对于 Vector&ArrayList、Hashtable&HashMap，要记住线程安全的问题，记住 Vector 与 Hashtable 是旧的，是 java 一诞生就提供了的，它们是线程安全的，ArrayList 与 HashMap 是 java2时才提供的，它们是线程不安全的。所以，我们讲课时先讲老的。

（2）数据增长：

ArrayList 与 Vector 都有一个初始的容量大小，当存储进它们里面的元素的个数超过了容量时，就需要增加 ArrayList 与 Vector 的存储空间，每次要增加存储空间时，不是只增加一个存储单元，而是增加多个存储单元，每次增加的存储单元的个数在内存空间利用与程序效率之间要取得一定的平衡。Vector 默认增长为原来两倍，而 ArrayList 的增长策略在文档中没有明确规定（从源代码看到的是增长为原来的1.5倍）。ArrayList 与 Vector 都可以设置初始的空间大小，Vector 还可以设置增长的空间大小，而 ArrayList 没有提供设置增长空间的方法。

总结：即 Vector 增长原来的一倍，ArrayList 增加原来的0.5倍。

60、HashMap 和 Hashtable 的区别

（条理上还需要整理，也是先说相同点，再说不同点）

HashMap 是 Hashtable 的轻量级实现（非线程安全的实现），他们都完成了 Map 接口，主要区别在于 HashMap 允许空（null）键值（key）,由于非线程安全，在只有一个线程访问的情况下，效率要高于 Hashtable。

HashMap 允许将 null 作为一个 entry 的 key 或者 value，而 Hashtable 不允许。

HashMap 把 Hashtable 的 contains 方法去掉了，改成 containsvalue 和 containsKey。因为 contains 方法容易让人引起误解。

Hashtable 继承自 Dictionary 类，而 HashMap 是 Java1.2引进的 Map interface 的一个实现。

最大的不同是，Hashtable 的方法是 Synchronize 的，而 HashMap 不是，在多个线程访问 Hashtable 时，不需要自己为它的方法实现同步，而 HashMap 就必须为之提供外同步。

Hashtable 和 HashMap 采用的 hash/rehash 算法都大概一样，所以性能不会有很大的差异。

就 HashMap 与 HashTable 主要从三方面来说。

一.历史原因:Hashtable 是基于陈旧的 Dictionary 类的，HashMap 是 Java 1.2引进的 Map 接口的一个实现

二.同步性:Hashtable 是线程安全的，也就是说是同步的，而 HashMap 是线程序不安全的，不是同步的

三.值：只有 HashMap 可以让你将空值作为一个表的条目的 key 或 value

61、List 和 Map 区别?

一个是存储单列数据的集合，另一个是存储键和值这样的双列数据的集合，List 中存储的数据是有顺序，并且允许重复；Map 中存储的数据是没有顺序的，其键是不能重复的，它的值是可以有重复的。

62、List, Set, Map 是否继承自 Collection 接口?

List，Set 是，Map 不是

63、List、Map、Set 三个接口，存取元素时，各有什么特点？

这样的题属于随意发挥题：这样的题比较考水平，两个方面的水平：一是要真正明白这些内容，二是要有较强的总结和表述能力。如果你明白，但表述不清楚，在别人那里则等同于不明白。

首先，List 与 Set 具有相似性，它们都是单列元素的集合，所以，它们有一个功共同的父接口，叫 Collection。Set 里面不允许有重复的元素，所谓重复，即不能有两个相等（注意，不是仅仅是相同）的对象，即假设 Set 集合中有了一个 A 对象，现在我要向 Set 集合再存入一个 B 对象，但 B 对象与 A 对象 equals 相等，则 B 对象存储不进去，所以，Set 集合的 add 方法有一个 boolean 的返回值，当集合中没有某个元素，此时 add 方法可成功加入该元素时，则返回 true，当集合含有与某个元素 equals 相等的元素时，此时 add 方法无法加入该元素，返回结果为 false。Set 取元素时，没法说取第几个，只能以 Iterator 接口取得所有的元素，再逐一遍历各个元素。

List 表示有先后顺序的集合，注意，不是那种按年龄、按大小、按价格之类的排序。当我们多次调用 add(Obj e)方法时，每次加入的对象就像火车站买票有排队顺序一样，按先来后到的顺序排序。有时候，也可以插队，即调用 add(int index,Obj e)方法，就可以指定当前对象在集合中的存放位置。一个对象可以被反复存储进 List 中，每调用一次 add 方法，这个对象就被插入进集合中一次，其实，并不是把这个对象本身存储进了集合中，而是在集合中用一个索引变量指向这个对象，当这个对象被 add 多次时，即相当于集合中有多个索引指向了这个对象，如图 x 所示。List 除了可以以 Iterator 接口取得所有的元素，再逐一遍历各个元素之外，还可以调用 get(index i)来明确说明取第几个。

Map 与 List 和 Set 不同，它是双列的集合，其中有 put 方法，定义如下：put(obj key,objvalue)，每次存储时，要存储一对 key/value，不能存储重复的 key，这个重复的规则也是按 equals 比较相等。取则可以根据 key 获得相应的 value，即 get(Object key)返回值为 key 所对应的 value。另外，也可以获得所有的 key 的结合，还可以获得所有的 value 的结合，还可以获得 key 和 value 组合成的 Map.Entry 对象的集合。

List 以特定次序来持有元素，可有重复元素。Set 无法拥有重复元素,内部排序。Map 保存 key-value 值，value 可多值。

HashSet 按照 hashcode 值的某种运算方式进行存储，而不是直接按 hashCode 值的大小进行存储。例如，"abc"---> 78，"def" ---> 62，"xyz" ---> 65在 hashSet 中的存储顺序不是 62,65,78，这些问题感谢以前一个叫崔健的学员提出，最后通过查看源代码给他解释清楚，看本次培训学员当中有多少能看懂源码。LinkedHashSet 按插入的顺序存储，那被存储对象的 hashcode 方法还有什么作用呢？学员想想!hashset 集合比较两个对象是否相等，首先看 hashcode 方法是否相等，然后看 equals 方法是否相等。new 两个 Student 插入到 HashSet 中，看 HashSet 的 size，实现 hashcode 和 equals 方法后再看 size。

同一个对象可以在 Vector 中加入多次。往集合里面加元素，相当于集合里用一根绳子连接到了目标对象。往 HashSet 中却加不了多次的。

64、说出 ArrayList,Vector, LinkedList 的存储性能和特性

ArrayList 和 Vector 都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector 由于使用了 synchronized 方法（线程安全），通常性能上较 ArrayList 差，而 LinkedList 使用双向链表实现存储，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快。

LinkedList 也是线程不安全的，LinkedList 提供了一些方法，使得 **LinkedList** 可以被当作堆栈和队列来使用。

65、去掉一个 Vector 集合中重复的元素

```
Vector newVector = new Vector();
For (int i=0;i<vector.size();i++)
{
Object obj = vector.get(i);
        if(!newVector.contains(obj);
                newVector.add(obj);
}
```

还有一种简单的方式，HashSet set = new HashSet(vector);

66、Collection 和 Collections 的区别。

Collection 是集合类的上级接口，继承与他的接口主要有 Set 和 List.

Collections 是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。

67、Set 里的元素是不能重复的，那么用什么方法来区分重复与否呢?是用==还是 equals()?它们有何区别?

Set 里的元素是不能重复的，元素重复与否是使用 equals()方法进行判断的。

equals()和==方法决定引用值是否指向同一对象 equals()在类中被覆盖，为的是当两个分离的对象的内容和类型相配的话，返回真值。

68、你所知道的集合类都有哪些？主要方法？

最常用的集合类是 List 和 Map。 List 的具体实现包括 ArrayList 和 Vector，它们是可变大小的列表，比较适合构建、存储和操作任何类型对象的元素列表。 List 适用于按数值索引访问元素的情形。

Map 提供了一个更通用的元素存储方法。 Map 集合类用于存储元素对（称作"键"和"值"），其中每个键映射到一个值。

ArrayList/VectoràList

àCollection

HashSet/TreeSetàSet

PropetiesàHashTable

àMap

Treemap/HashMap

我记的不是方法名，而是思想，我知道它们都有增删改查的方法，但这些方法的具体名称，我记得不是很清楚，对于 set，大概的方法是 add,remove, contains；对于 map，大概的方法就是 put,remove，contains 等，因为，我只要在 eclispe 下按点操作符，很自然的这些方法就出来了。我记住的一些思想就是 List 类会有 get(int index)这样的方法，因为它可以按顺序取元素，而 set 类中没有 get(int index)这样的方法。List 和 set 都可以迭代出所有元素，迭代时先要得到一个 iterator 对象，所以，set 和 list 类都有一个 iterator 方法，用于返回那个 iterator 对象。map 可以返回三个集合，一个是返回所有的 key 的集合，另外一个返回的是所有 value 的集合，再一个返回的 key 和 value 组合成的 EntrySet 对象的集合，map 也有 get 方法，参数是 key，返回值是 key 对应的 value。

69、两个对象值相同**(x.equals(y) == true)**，但却可有不同的 **hash code**，这句话对不对**?**

对。

如果对象要保存在 HashSet 或 HashMap 中，它们的 equals 相等，那么，它们的 hashcode 值就必须相等。

如果不是要保存在 HashSet 或 HashMap，则与 hashcode 没有什么关系了，这时候 hashcode 不等是可以的，例如 arrayList 存储的对象就不用实现 hashcode，当然，我们没有理由不实现，通常都会去实现的。

70、**TreeSet** 里面放对象，如果同时放入了父类和子类的实例对象，那比较时使用的是父类的 **compareTo** 方法，还是使用的子类的 **compareTo** 方法，还是抛异常！

（应该是没有针对问题的确切的答案，当前的 add 方法放入的是哪个对象，就调用哪个对象的 compareTo 方法，至于这个 compareTo 方法怎么做，就看当前这个对象的类中是如何编写这个方法的）

实验代码：

```
public class ParentimplementsComparable {
    private int age = 0;
    public Parent(int age){
        this.age = age;
    }
    public int compareTo(Object o){
        // TODO Auto-generated method stub
        System.out.println("method ofparent");
        Parent o1 = (Parent)o;
        return age>o1.age?1:age<o1.age?-1:0;
    }


}


public class Childextends Parent {
```

```
    public Child(){
        super(3);
    }
    public int compareTo(Object o){

            // TODO Auto-generated methodstub
            System.out.println("methodof child");
//          Child o1 = (Child)o;
            return 1;

    }
}

public class TreeSetTest {

    /**
     * @paramargs
     */
    public static voidmain(String[] args) {
        // TODO Auto-generated method stub
        TreeSet set = new TreeSet();
        set.add(newParent(3));
        set.add(new Child());
        set.add(newParent(4));
        System.out.println(set.size());
    }

}
```

71、说出一些常用的类，包，接口，请各举 5 个

要让人家感觉你对 java ee 开发很熟，所以，不能仅仅只列 core java 中的那些东西，要多列你在做 ssh 项目中涉及的那些东西。就写你最近写的那些程序中涉及的那些类。

常用的类：BufferedReader BufferedWriter FileReader FileWirter String Integer java.util.Date，System，Class，List,HashMap

常用的包：java.lang java.io java.util java.sql,javax.servlet,org.apache.strtuts.action,org.hibernate

常用的接口：Remote List Map Document NodeList,Servlet,HttpServletRequest,HttpServletResponse,Transaction(Hibernate)、Session(Hibernate),HttpSession

72、java 中有几种类型的流？JDK 为每种类型的流提供了一些抽象类以供继承，请说出他们分别是哪些类？

字节流，字符流。字节流继承于 InputStream OutputStream，字符流继承于 InputStreamReaderOutputStreamWriter。在 java.io 包中还有许多其他的流，主要是为了提高性能和使用方便。

73、字节流与字符流的区别

要把一片二进制数据数据逐一输出到某个设备中，或者从某个设备中逐一读取一片二进制数据，不管输入输出设备是什么，我们要用统一的方式来完成这些操作，用一种抽象的方式进行描述，这个抽象描述方式起名为 IO 流，对应的抽象类为 OutputStream 和 InputStream，不同的实现类就代表不同的输入和输出设备，它们都是针对字节进行操作的。

在应用中，经常要完全是字符的一段文本输出去或读进来，用字节流可以吗？计算机中的一切最终都是二进制的字节形式存在。对于"中国"这些字符，首先要得到其对应的字节，然后将字节写入到输出流。读取时，首先读到的是字节，可是我们要把它显示为字符，我们需要将字节转换成字符。由于这样的需求很广泛，人家专门提供了字符流的包装类。

底层设备永远只接受字节数据，有时候要写字符串到底层设备，需要将字符串转成字节再进行写入。字符流是字节流的包装，字符流则是直接接受字符串，它内部将串转成字节，再写入底层设备，这为我们向 IO 设别写入或读取字符串提供了一点点方便。

字符向字节转换时，要注意编码的问题，因为字符串转成字节数组，

其实是转成该字符的某种编码的字节形式，读取也是反之的道理。

讲解字节流与字符流关系的代码案例：

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.io.PrintWriter;

public class IOTest {
    public static void main(String[]args) throws Exception {
        String str = "中国人";
        /*FileOutputStreamfos   = newFileOutputStream("1.txt");

        fos.write(str.getBytes("UTF-8"));
        fos.close();*/

        /*FileWriter fw =new FileWriter("1.txt");
        fw.write(str);
        fw.close();*/
        PrintWriter pw =new PrintWriter("1.txt","utf-8");
        pw.write(str);
        pw.close();

        /*FileReader fr =new FileReader("1.txt");
        char[] buf = newchar[1024];
        int len =fr.read(buf);
        String myStr = newString(buf,0,len);
        System.out.println(myStr);*/
        /*FileInputStreamfr = new FileInputStream("1.txt");
        byte[] buf = newbyte[1024];
```

```
        int len =fr.read(buf);
        String myStr = newString(buf,0,len,"UTF-8");
        System.out.println(myStr);*/
        BufferedReader br =new BufferedReader(
                    newInputStreamReader(
                          newFileInputStream("1.txt"),"UTF-8"
                          )
                    );
        String myStr =br.readLine();
        br.close();
        System.out.println(myStr);
    }

}
```

74、什么是 **java** 序列化，如何实现 **java** 序列化？或者请解释 **Serializable** 接口的作用。

我们有时候将一个 java 对象变成字节流的形式传出去或者从一个字节流中恢复成一个 java 对象，例如，要将 java 对象存储到硬盘或者传送给网络上的其他计算机，这个过程我们可以自己写代码去把一个 java 对象变成某个格式的字节流再传输，但是，jre 本身就提供了这种支持，我们可以调用 OutputStream 的 writeObject 方法来做，如果要让 java 帮我们做，要被传输的对象必须实现 **serializable** 接口，这样，javac 编译时就会进行特殊处理，编译的类才可以被 writeObject 方法操作，这就是所谓的序列化。需要被序列化的类必须实现 Serializable 接口，该接口是一个 mini 接口，其中没有需要实现的方法，implementsSerializable 只是为了标注该对象是可被序列化的。

例如，在 web 开发中，如果对象被保存在了 Session 中，tomcat 在重启时要把 Session 对象序列化到硬盘，这个对象就必须实现 Serializable 接口。如果对象要经过分布式系统进行网络传输或通过 rmi 等远程调用，这就需要在网络上传输对象，被传输的对象就必须实现 Serializable 接口。

75、描述一下 JVM 加载 class 文件的原理机制?

JVM 中类的装载是由 ClassLoader 和它的子类来实现的,Java ClassLoader 是一个重要的 Java 运行时系统组件。它负责在运行时查找和装入类文件的类。

76、heap 和 stack 有什么区别。

java 的内存分为两类，一类是栈内存，一类是堆内存。栈内存是指程序进入一个方法时，会为这个方法单独分配一块私属存储空间，用于存储这个方法内部的局部变量，当这个方法结束时，分配给这个方法的栈会释放，这个栈中的变量也将随之释放。

堆是与栈作用不同的内存，一般用于存放不放在当前方法栈中的那些数据，例如，使用 new 创建的对象都放在堆里，所以，它不会随方法的结束而消失。方法中的局部变量使用 **final** 修饰后，放在堆中，而不是栈中。

77、GC 是什么?为什么要有 GC?

GC 是垃圾收集的意思（Gabage Collection）,内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java 提供的 GC 功能可以自动监测对象是否超过作用域从而达到自动回收内存的目的，Java 语言没有提供释放已分配内存的显示操作方法。

78、垃圾回收的优点和原理。并考虑 2 种回收机制。

Java 语言中一个显著的特点就是引入了垃圾回收机制，使 c++程序员最头疼的内存管理的问题迎刃而解，它使得 Java 程序员在编写程序的时候不再需要考虑内存管理。由于有个垃圾回收机制，Java 中的对象不再有"作用域"的概念，只有对象的引用才有"作用域"。垃圾回收可以有效的防止内存泄露，有效的使用可以使用的内存。垃圾回收器通常是作为一个单独的低级别的线程运行，不可预知的情况下对内存堆中已经死亡的或者长时间没有使用的对象进行清楚和回收，程序员不能实时的调用垃圾回收器对某个对象或所有对象进行垃圾回收。回收机制有分代复制垃圾回收和标记垃圾回收，增量垃圾回收。

79、垃圾回收器的基本原理是什么？垃圾回收器可以马上回收内存吗？有什么办法主动通知虚拟机进行垃圾回收？

对于 GC 来说，当程序员创建对象时，GC 就开始监控这个对象的地址、大小以及使用情况。通常，GC 采用有向图的方式记录和管理堆(heap)中的所有对象。通过这种方式确定哪些对

象是"可达的"，哪些对象是"不可达的"。当 GC 确定一些对象为"不可达"时，GC 就有责任回收这些内存空间。可以。程序员可以手动执行 **System.gc()**，通知 GC 运行，但是 Java 语言规范并不保证 GC 一定会执行。

80、什么时候用 assert。

assertion(断言)在软件开发中是一种常用的调试方式，很多开发语言中都支持这种机制。在实现中，assertion 就是在程序中的一条语句，它对一个 boolean 表达式进行检查，一个正确程序必须保证这个 boolean 表达式的值为 true；如果该值为 false，说明程序已经处于不正确的状态下，assert 将给出警告或退出。一般来说，**assertion** 用于保证程序最基本、关键的正确性。assertion 检查通常在开发和测试时开启。为了提高性能，在软件发布后，assertion 检查通常是关闭的。

```
package com.huawei.interview;

publicclass AssertTest {

    /**
     * @paramargs
     */
    public static voidmain(String[] args) {
        // TODO Auto-generated method stub
        int i = 0;
        for(i=0;i<5;i++)
        {
            System.out.println(i);
        }
        //假设程序不小心多了一句--i;
        --i;
        assert i==5;
    }
```

}

81、java 中会存在内存泄漏吗，请简单描述。

所谓内存泄露就是指一个不再被程序使用的对象或变量一直被占据在内存中。java 中有垃圾回收机制，它可以保证一对象不再被引用的时候，即对象编程了孤儿的时候，对象将自动被垃圾回收器从内存中清除掉。由于 Java 使用有向图的方式进行垃圾回收管理，可以消除引用循环的问题，例如有两个对象，相互引用，只要它们和根进程不可达的，那么 GC 也是可以回收它们的，例如下面的代码可以看到这种情况的内存回收：

```
package com.huawei.interview;

import java.io.IOException;

publicclass GarbageTest {

    /**
     * @paramargs
     * @throwsIOException
     */
    public static voidmain(String[] args)throws IOException {
        // TODO Auto-generated method stub
        try {
            gcTest();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        System.out.println("hasexited gcTest!");
        System.in.read();
        System.in.read();
        System.out.println("out begingc!");
        for(int i=0;i<100;i++)
```

```
    {
        System.gc();
        System.in.read();
        System.in.read();
    }
}

private static voidgcTest()throws IOException {
    System.in.read();
    System.in.read();
    Person p1 = new Person();
    System.in.read();
    System.in.read();
    Person p2 = new Person();
    p1.setMate(p2);
    p2.setMate(p1);
    System.out.println("beforeexit gctest!");
    System.in.read();
    System.in.read();
    System.gc();
    System.out.println("exitgctest!");
}

private static classPerson
{
    byte[] data =new byte[20000000];
    Person mate = null;
    public void setMate(Personother)
    {
        mate = other;
    }
```

```
    }
}
```

java 中的内存泄露的情况：长生命周期的对象持有短生命周期对象的引用就很可能发生内存泄露，尽管短生命周期对象已经不再需要，但是因为长生命周期对象持有它的引用而导致不能被回收，这就是 java 中内存泄露的发生场景，通俗地说，就是程序员可能创建了一个对象，以后一直不再使用这个对象，这个对象却一直被引用，即这个对象无用但是却无法被垃圾回收器回收的，这就是 java 中可能出现内存泄露的情况，例如，缓存系统，我们加载了一个对象放在缓存中(例如放在一个全局 map 对象中)，然后一直不再使用它，这个对象一直被缓存引用，但却不再被使用。

检查 java 中的内存泄露，一定要让程序将各种分支情况都完整执行到程序结束，然后看某个对象是否被使用过，如果没有，则才能判定这个对象属于内存泄露。

如果一个外部类的实例对象的方法返回了一个内部类的实例对象，这个内部类对象被长期引用了，即使那个外部类实例对象不再被使用，但由于内部类持久外部类的实例对象，这个外部类对象将不会被垃圾回收，这也会造成内存泄露。

下面内容来自于网上（主要特点就是清空堆栈中的某个元素，并不是彻底把它从数组中拿掉，而是把存储的总数减少，本人写得可以比这个好，在拿掉某个元素时，顺便也让它从数组中消失，将那个元素所在的位置的值设置为 null 即可）：

我实在想不到比那个堆栈更经典的例子了,以致于我还要引用别人的例子，下面的例子不是我想到的，是书上看到的，当然如果没有在书上看到，可能过一段时间我自己也想的到，可是那时我说是我自己想到的也没有人相信的。

```
public class Stack {
 private Object[] elements=new Object[10];
 private int size = 0;
 public void push(Object e){
 ensureCapacity();
 elements[size++] = e;
 }
 public Object pop(){
 if( size == 0)
throw new EmptyStackException();
 return elements[--size];
```

```
}
private void ensureCapacity(){
if(elements.length == size){
Object[] oldElements = elements;
elements = new Object[2 * elements.length+1];
System.arraycopy(oldElements,0, elements, 0, size);
}
}
}
```

上面的原理应该很简单，假如堆栈加了10个元素，然后全部弹出来，虽然堆栈是空的，没有我们要的东西，但是这是个对象是无法回收的，这个才符合了内存泄露的两个条件：无用，无法回收。

但是就是存在这样的东西也不一定会导致什么样的后果，如果这个堆栈用的比较少，也就浪费了几个 K 内存而已，反正我们的内存都上 G 了，哪里会有什么影响，再说这个东西很快就会被回收的，有什么关系。下面看两个例子。

例子1

```
public class Bad{
public static Stack s=Stack();
static{
s.push(new Object());
s.pop(); //这里有一个对象发生内存泄露
s.push(new Object()); //上面的对象可以被回收了，等于是自愈了
}
}
```

因为是 static，就一直存在到程序退出，但是我们也可以看到它有自愈功能，就是说如果你的 Stack 最多有100个对象，那么最多也就只有100个对象无法被回收其实这个应该很容易理解，Stack 内部持有100个引用，最坏的情况就是他们都是无用的，因为我们一旦放新的进取，以前的引用自然消失！

内存泄露的另外一种情况：当一个对象被存储进 HashSet 集合中以后，就不能修改这个对象中的那些参与计算哈希值的字段了，否则，对象修改后的哈希值与最初存储进 HashSet 集合中时的哈希值就不同了，在这种情况下，即使在 contains 方法使用该对象的当前引用作为的参数去 HashSet 集合中检索对象，也将返回找不到对象的结果，这也会导致无法从 HashSet 集合中单独删除当前对象，造成内存泄露。

82、能不能自己写个类，也叫 java.lang.String？

可以，但在应用的时候，需要用自己的类加载器去加载，否则，系统的类加载器永远只是去加载 jre.jar 包中的那个 java.lang.String。由于在 tomcat 的 web 应用程序中，都是由 webapp 自己的类加载器先自己加载 WEB-INF/classess 目录中的类，然后才委托上级的类加载器加载，如果我们在 tomcat 的 web 应用程序中写一个 java.lang.String，这时候 Servlet 程序加载的就是我们自己写的 java.lang.String，但是这么干就会出很多潜在的问题，原来所有用了 java.lang.String 类的都将出现问题。

虽然 java 提供了 **endorsed** 技术，可以覆盖 jdk 中的某些类，具体做法是….。但是，能够被覆盖的类是有限制范围，反正不包括 java.lang 这样的包中的类。

（下面的例如主要是便于大家学习理解只用，不要作为答案的一部分，否则，人家怀疑是题目泄露了）例如，运行下面的程序：

```
package java.lang;

publicclass String {

    /**
     * @paramargs
     */
    public static voidmain(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("string");
    }

}
```

报告的错误如下：

java.lang.NoSuchMethodError:main

Exception inthread "main"

这是因为加载了 jre 自带的 java.lang.String，而该类中没有 main 方法。

83. Java 代码查错

1.

```
abstract class Name {
    private String name;
    public abstract boolean isStupidName(String name) {}
}
```

大侠们，这有何错误?

答案: 错。abstract method 必须以分号结尾，且不带花括号。

2.

```
public class Something {
    void doSomething () {
        private String s = "";
        int l = s.length();
    }
}
```

有错吗?

答案: 错。局部变量前不能放置任何访问修饰符 (private，public，和 protected)。final 可以用来修饰局部变量

(final 如同 abstract 和 strictfp，都是非访问修饰符，strictfp 只能修饰 class 和 method 而非 variable)。

3.

```
abstract class Something {
    private abstract String doSomething ();
}
```

这好像没什么错吧?

答案: 错。abstract 的 methods 不能以 private 修饰。abstract 的 methods 就是让子类 implement(实现)具体细节的，怎么可以用 private 把 abstract

method 封锁起来呢? (同理，abstract method 前不能加 final)。

4.

```
public class Something {
    public int addOne(final int x) {
```

```
        return ++x;
    }
}
```

这个比较明显。

答案: 错。int x 被修饰成 final，意味着 x 不能在 addOne method 中被修改。

5.

```
public class Something {
    public static void main(String[] args) {
        Other o = new Other();
        new Something().addOne(o);
    }
    public void addOne(final Other o) {
        o.i++;
    }
}
class Other {
    public int i;
}
```

和上面的很相似，都是关于 final 的问题，这有错吗?

答案: 正确。在 addOne method 中，参数 o 被修饰成 final。如果在 addOne method 里我们修改了 o 的 reference

(比如: o = new Other();)，那么如同上例这题也是错的。但这里修改的是 o 的 member vairable

(成员变量)，而 o 的 reference 并没有改变。

6.

```
class Something {
     int i;
     public void doSomething() {
          System.out.println("i = "+ i);
     }
}
```

有什么错呢? 看不出来啊。

答案: 正确。输出的是"i = 0"。int i 属於 instant variable (实例变量，或叫成员变量)。instant variable 有 default value。int 的 default value 是0。

7.

```
class Something {
    final int i;
    public void doSomething() {
        System.out.println("i = "+ i);
    }
}
```

和上面一题只有一个地方不同，就是多了一个 final。这难道就错了吗?

答案: 错。final int i 是个 final 的 instant variable (实例变量，或叫成员变量)。final 的 instant variable 没有 default value，必须在 constructor (构造器)结束之前被赋予一个明确的值。可以修改为"final int i =0;"。

8.

```
public class Something {
     public static void main(String[] args) {
        Something s = new Something();
       System.out.println("s.doSomething() returns " + doSomething());
    }
    public String doSomething() {
        return "Do something ...";
    }
}
```

看上去很完美。

答案: 错。看上去在 main 里 call doSomething 没有什么问题，毕竟两个 methods 都在同一个 class 里。但仔细看，main 是 static 的。static method 不能直接 call non-staticmethods。可改成"System.out.println("s.doSomething()returns " + s.doSomething());"。同理，static method 不能访问 non-static instant variable。

9.

此处，Something 类的文件名叫 OtherThing.java

```
class Something {
    private static void main(String[] something_to_do){
        System.out.println("Dosomething ...");
    }
}
```

这个好像很明显。

答案: 正确。从来没有人说过 Java 的 Class 名字必须和其文件名相同。但 public class 的名字必须和文件名相同。

10．

```
interface   A{
    int x = 0;
}
class B{
    int x =1;
}
class C extends B implements A {
    public void pX(){
        System.out.println(x);
    }
    public static void main(String[] args) {
        new C().pX();
    }
}
```

答案：错误。在编译时会发生错误(错误描述不同的 JVM 有不同的信息，意思就是未明确的 x 调用，两个 x 都匹配（就象在同时 import java.util 和 java.sql 两个包时直接声明 Date 一样）。对于父类的变量,可以用 super.x 来明确，而接口的属性默认隐含为 public staticfinal.所以可以通过 A.x 来明确。

11.

```
interface Playable {
    void play();
}
interface Bounceable {
    void play();
}
interface Rollable extends Playable, Bounceable {
    Ball ball = new Ball("PingPang");
}
class Ball implements Rollable {
    private String name;
    public String getName() {
        return name;
    }
    public Ball(String name) {
```

```
            this.name =name;
    }
    public void play() {
            ball = newBall("Football");
            System.out.println(ball.getName());
    }
}
```

这个错误不容易发现。

答案: 错。"interfaceRollable extends Playable, Bounceable"没有问题。interface 可继承多个 interfaces，所以这里没错。问题出在 interface Rollable 里的"Ball ball =new Ball("PingPang");"。任何在 interface 里声明的 interface variable (接口变量，也可称成员变量)，默认为 public static final。也就是说"Ball ball = new Ball("PingPang");"实际上是"public staticfinal Ball ball = new Ball("PingPang");"。在 Ball 类的 Play()方法中，"ball = newBall("Football");"改变了 ball 的 reference，而这里的 ball 来自 Rollable interface，Rollable interface 里的 ball 是 public static final 的，final 的 object 是不能被改变 reference 的。因此编译器将在"ball = newBall("Football");"这里显示有错。

二.算法与编程

1、编写一个程序，将 **a.txt** 文件中的单词与 **b.txt** 文件中的单词交替合并到 **c.txt** 文件中，**a.txt** 文件中的单词用回车符分隔，**b.txt** 文件中用回车或空格进行分隔。

答：

```
packagecn.itcast;

import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;

public class MainClass{
        public static voidmain(String[] args) throws Exception{
                FileManager a= new FileManager("a.txt",new char[]{'\n'});
                FileManager b= new FileManager("b.txt",new char[]{'\n',' '});
                FileWriter c= new FileWriter("c.txt");
                String aWord= null;
```

```
        String bWord= null;
        while((aWord= a.nextWord()) !=null ){
            c.write(aWord+ "\n");
            bWord= b.nextWord();
            if(bWord!= null)
                c.write(bWord+ "\n");
        }

        while((bWord= b.nextWord()) != null){
            c.write(bWord+ "\n");
        }
        c.close();
    }

}


class FileManager{

    String[] words =null;
    int pos = 0;
    publicFileManager(String filename,char[] seperators) throws Exception{
        File f = newFile(filename);
        FileReaderreader = new FileReader(f);
        char[] buf =new char[(int)f.length()];
        int len =reader.read(buf);
        Stringresults = new String(buf,0,len);
        String regex= null;
        if(seperators.length>1 ){
            regex= "" + seperators[0] + "|" + seperators[1];
        }else{
```

```
                regex= "" + seperators[0];
            }
            words =results.split(regex);
        }

        public StringnextWord(){
            if(pos ==words.length)
                returnnull;
            returnwords[pos++];
        }

}
```

2、编写一个程序，将 **d:\java** 目录下的所有**.java** 文件复制到 **d:\jad** 目录下，并将原来文件的扩展名从**.java** 改为**.jad**。

（大家正在做上面这道题，网上迟到的朋友也请做做这道题，找工作必须能编写这些简单问题的代码！）

答：**listFiles** 方法接受一个 **FileFilter** 对象，这个 **FileFilter** 对象就是过虑的策略对象，不同的人提供不同的 **FileFilter** 实现，即提供了不同的过滤策略。

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FilenameFilter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class Jad2Java {

        public static voidmain(String[] args) throws Exception {
            File srcDir =new File("java");
```

```
		if(!(srcDir.exists()&& srcDir.isDirectory()))
				thrownew Exception("目录不存在");
		File[] files= srcDir.listFiles(
			newFilenameFilter(){

					publicboolean accept(File dir, String name) {
						returnname.endsWith(".java");
					}

				}
		);

		System.out.println(files.length);
		File destDir= new File("jad");
		if(!destDir.exists())destDir.mkdir();
		for(File f:files){
			FileInputStream    fis = new FileInputStream(f);
			StringdestFileName = f.getName().replaceAll("\\.java$", ".jad");
			FileOutputStreamfos = new FileOutputStream(new
File(destDir,destFileName));
			copy(fis,fos);
			fis.close();
			fos.close();
		}
	}

	private static voidcopy(InputStream ips,OutputStream ops) throws Exception{
		int len = 0;
		byte[] buf =new byte[1024];
		while((len =ips.read(buf)) != -1){
			ops.write(buf,0,len);
```

}

}

}

由本题总结的思想及策略模式的解析：

1.

class jad2java{

1. 得到某个目录下的所有的 java 文件集合

1.1 得到目录 File srcDir = newFile("d:\\java");

1.2 得到目录下的所有 java 文件：File[] files =srcDir.listFiles(new MyFileFilter());

1.3 只想得到.java 的文件： class MyFileFilterimplememyts FileFilter{

publicboolean accept(File pathname){

returnpathname.getName().endsWith(".java")

}

}

2.将每个文件复制到另外一个目录，并改扩展名

2.1 得到目标目录，如果目标目录不存在，则创建之

2.2 根据源文件名得到目标文件名，注意要用正则表达式，注意.的转义。

2.3 根据表示目录的 File 和目标文件名的字符串，得到表示目标文件的 File。

//要在硬盘中准确地创建出一个文件，需要知道文件名和文件的目录。

2.4 将源文件的流拷贝成目标文件流，拷贝方法独立成为一个方法，方法的参数采用抽象流的形式。

//方法接受的参数类型尽量面向父类，越抽象越好，这样适应面更宽广。

}

分析 listFiles 方法内部的策略模式实现原理

```
File[] listFiles(FileFilter filter){
        File[] files =listFiles();
        //ArraylistacceptedFilesList = new ArrayList();
        File[] acceptedFiles= new File[files.length];
        int pos = 0;
        for(File file:files){
                booleanaccepted = filter.accept(file);
                if(accepted){
                        //acceptedFilesList.add(file);
                        acceptedFiles[pos++]= file;
                }
        }

        Arrays.copyOf(acceptedFiles,pos);
        //return(File[])accpetedFilesList.toArray();

}
```

3、编写一个截取字符串的函数，输入为一个字符串和字节数，输出为按字节截取的字符串，但要保证汉字不被截取半个，如"我 **ABC**"，**4**，应该截取"我 **AB**"，输入"我 **ABC** 汉 **DEF**"，**6**，应该输出"我 **ABC**"，而不是"我 **ABC+**汉的半个"。

答：

首先要了解中文字符有多种编码及各种编码的特征。

假设 n 为要截取的字节数。

```
public static voidmain(String[] args) throws Exception{
        String str ="我 a 爱中华 abc 我爱传智 def';
        String str ="我 ABC 汉";
        int num =trimGBK(str.getBytes("GBK"),5);
        System.out.println(str.substring(0,num));
}
```

```
public staticint   trimGBK(byte[] buf,int n){
        int num = 0;
        booleanbChineseFirstHalf = false;
        for(inti=0;i<n;i++)
        {
                if(buf[i]<0&& !bChineseFirstHalf){
                        bChineseFirstHalf= true;
                }else{
                        num++;
                        bChineseFirstHalf= false;
                }
        }
        return num;
}
```

4、有一个字符串，其中包含中文字符、英文字符和数字字符，请统计和打印出各个字符的个数。

答：哈哈，其实包含中文字符、英文字符、数字字符原来是出题者放的烟雾弹。

```
String content = "中国 aadf 的111萨 bbb 菲的 zz 萨菲";
HashMap map = new HashMap();
for(int i=0;i<content.length;i++)
{
        char c =content.charAt(i);
        Integer num =map.get(c);
        if(num == null)
                num = 1;
        else
                num = num +1;
        map.put(c,num);
}
for(Map.EntrySet entry : map)
```

```
{
        system.out.println(entry.getkey()+ “:” + entry.getValue());
}
```

估计是当初面试的那个学员表述不清楚，问题很可能是：

如果一串字符如"aaaabbc 中国1512"要分别统计英文字符的数量，中文字符的数量，和数字字符的数量，假设字符中没有中文字符、英文字符、数字字符之外的其他特殊字符。

```
int engishCount;
int chineseCount;
int digitCount;
for(int i=0;i<str.length;i++)
{
    charch = str.charAt(i);
    if(ch>=’0’&& ch<=’9’)
    {
        digitCount++
    }
    elseif((ch>=’a’&& ch<=’z’) || (ch>=’A’ && ch<=’Z’))
    {
        engishCount++;
    }
    else
    {
        chineseCount++;
    }
}
System.out.println(……………);
```

5、说明生活中遇到的二叉树，用 java 实现二叉树

这是组合设计模式。

我有很多个(假设10万个)数据要保存起来，以后还需要从保存的这些数据中检索是否存在某个数据，（我想说出二叉树的好处，该怎么说呢？那就是说别人的缺点），假如存在数组中，

那么，碰巧要找的数字位于99999那个地方，那查找的速度将很慢，因为要从第1个依次往后取，取出来后进行比较。平衡二叉树（构建平衡二叉树需要先排序，我们这里就不作考虑了）可以很好地解决这个问题，但二叉树的遍历（前序，中序，后序）效率要比数组低很多，原理如下图：

代码如下：

```
package com.huawei.interview;

publicclass Node {
    public int value;
    public Node left;
    public Node right;

    public void store(intvalue)
    {
        if(value<this.value)
        {
            if(left ==null)
            {
                left = new Node();
                left.value=value;
            }
            else
            {
                left.store(value);
            }
        }
        else if(value>this.value)
        {
            if(right ==null)
            {
                right = new Node();
```

```
                right.value=value;
            }
            else
            {
                right.store(value);
            }
        }
    }

    public boolean find(intvalue)
    {
        System.out.println("happen" +this.value);
        if(value ==this.value)
        {
            return true;
        }
        else if(value>this.value)
        {
            if(right ==null)returnfalse;
            return right.find(value);
        }else
        {
            if(left ==null)returnfalse;
            return left.find(value);
        }

    }

    public   void preList()
    {
        System.out.print(this.value+ ",");
```

```
    if(left!=null)left.preList();
    if(right!=null) right.preList();
}

public void middleList()
{
    if(left!=null)left.preList();
    System.out.print(this.value+ ",");
    if(right!=null)right.preList();
}
public void afterList()
{
    if(left!=null)left.preList();
    if(right!=null)right.preList();
    System.out.print(this.value+ ",");
}
public static voidmain(String [] args)
{
    int [] data =new int[20];
    for(inti=0;i<data.length;i++)
    {
        data[i] = (int)(Math.random()*100)+ 1;
        System.out.print(data[i] +",");
    }
    System.out.println();

    Node root = new Node();
    root.value = data[0];
    for(inti=1;i<data.length;i++)
    {
        root.store(data[i]);
```

```
        }

        root.find(data[19]);

        root.preList();
        System.out.println();
        root.middleList();
        System.out.println();
        root.afterList();
    }
}
```

----------------又一次临场写的代码---------------------------

```
importjava.util.Arrays;
importjava.util.Iterator;

public class Node{
    private Node left;
    private Node right;
    private int value;
    //private int num;

    public Node(int value){
        this.value = value;
    }
    public void add(int value){

        if(value > this.value)
        {
            if(right != null)
                right.add(value);
            else
```

```
            {
                Node node = new Node(value);
                right = node;
            }
        }
        else{
            if(left != null)
                left.add(value);
            else
            {
                Node node = new Node(value);
                left = node;
            }
        }
    }

    public boolean find(int value){
        if(value == this.value) return true;
        else if(value > this.value){
            if(right == null) return false;
            else return right.find(value);
        }else{
            if(left == null) return false;
            else return left.find(value);
        }

    }

    public void display(){
        System.out.println(value);
        if(left != null) left.display();
```

```java
        if(right != null) right.display();

    }

    /*public Iterator iterator(){

    }*/

    public static void main(String[] args){
        int[] values = new int[8];
        for(int i=0;i<8;i++){
            int num = (int)(Math.random() * 15);
            //System.out.println(num);
            //if(Arrays.binarySearch(values,num)<0)
            if(!contains(values,num))
                values[i] = num;
            else
                i--;
        }

        System.out.println(Arrays.toString(values));

        Node root = new Node(values[0]);
        for(int i=1;i<values.length;i++){
            root.add(values[i]);
        }

        System.out.println(root.find(13));

        root.display();
```

```
    }

    public static boolean contains(int [] arr,int value){
        int i = 0;
        for(;i<arr.length;i++){
            if(arr[i] == value) return true;

        }
        return false;
    }

}
```

6、从类似如下的文本文件中读取出所有的姓名，并打印出重复的姓名和重复的次数，并按重复次数排序：

1,张三,28

2,李四,35

3,张三,28

4,王五,35

5,张三,28

6,李四,35

7,赵六,28

8,田七,35

程序代码如下（答题要博得用人单位的喜欢，包名用该公司，面试前就提前查好该公司的网址，如果查不到，现场问也是可以的。还要加上实现思路的注释）：

```
package com.huawei.interview;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
```

```
import java.io.InputStreamReader;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.TreeSet;


publicclass GetNameTest {

    /**
     * @paramargs
     */
    public static voidmain(String[] args) {
        // TODO Auto-generated method stub
        //InputStream ips
=GetNameTest.class.getResourceAsStream("/com/huawei/interview/info.txt");
        //用上一行注释的代码和下一行的代码都可以，因为 info.txt 与 GetNameTest 类在
同一包下面，所以，可以用下面的相对路径形式

        Map results = new HashMap();
        InputStream ips = GetNameTest.class.getResourceAsStream("info.txt");
        BufferedReader in = newBufferedReader(new InputStreamReader(ips));
        String line = null;
        try {
            while((line=in.readLine())!=null)
            {
                dealLine(line,results);
            }
            sortResults(results);
        } catch (IOException e) {
```

```
            // TODO Auto-generated catchblock
            e.printStackTrace();
        }
    }

    static class User
    {
        public   String name;
        public Integer value;
        public User(String name,Integervalue)
        {
            this.name = name;
            this.value = value;
        }

        @Override
        public booleanequals(Object obj) {
            // TODO Auto-generated methodstub

            //下面的代码没有执行，说明往 treeset 中增加数据时，不会使用到 equals 方
法。
            boolean result =super.equals(obj);
            System.out.println(result);
            return result;
        }
    }

    private static voidsortResults(Map results) {
        // TODO Auto-generated method stub
        TreeSet sortedResults =newTreeSet(
                new Comparator(){
```

```
            public intcompare(Object o1, Object o2) {
                // TODOAuto-generated method stub
                User user1 = (User)o1;
                User user2 = (User)o2;
                /*如果 compareTo 返回结果0，则认为两个对象相等，新的对象不
会增加到集合中去
                 * 所以，不能直接用下面的代码，否则，那些个数相同的其他姓名
就打印不出来。
                 * */

                //returnuser1.value-user2.value;

//returnuser1.value<user2.value?-1:user1.value==user2.value?0:1;
                if(user1.value<user2.value)
                {
                    return -1;
                }else if(user1.value>user2.value)
                {
                    return 1;
                }else
                {
                    returnuser1.name.compareTo(user2.name);
                }
            }

        }
    );
    Iterator iterator =results.keySet().iterator();
    while(iterator.hasNext())
    {
        String name = (String)iterator.next();
```

```
            Integer value =(Integer)results.get(name);
            if(value > 1)
            {
                sortedResults.add(newUser(name,value));
            }
        }

        printResults(sortedResults);
    }
    private static voidprintResults(TreeSet sortedResults)
    {
        Iterator iterator    = sortedResults.iterator();
        while(iterator.hasNext())
        {
            User user = (User)iterator.next();
            System.out.println(user.name +":" + user.value);
        }
    }
    public static voiddealLine(String line,Map map)
    {
        if(!"".equals(line.trim()))
        {
            String [] results =line.split(",");
            if(results.length == 3)
            {
                String name = results[1];
                Integer value =(Integer)map.get(name);
                if(value ==null)value = 0;
                map.put(name,value + 1);
            }
        }
```

```
    }

}
```

7、写一个 Singleton 出来。

第一种：饱汉模式

```
public classSingleTon {
    private SingleTon(){
        }

    //实例化放在静态代码块里可提高程序的执行效率，但也可能更占用空间
    private final static SingleTon instance =new SingleTon();
    public static SingleTon getInstance(){
        return instance;
    }
}
```

第二种：饥汉模式

```
public classSingleTon {
    private SingleTon(){}

    private static instance = null;//newSingleTon();

    public static synchronized SingleTongetInstance(){
        if(instance == null)
            instance = new SingleTon();
        return instance;
    }
}
```

第三种：用枚举

```
public enum SingleTon{
    ONE;

}
```

第三：更实际的应用（在什么情况用单例）

```
public classSequenceGenerator{
    //下面是该类自身的业务功能代码
    private int count = 0;

    public synchronized int getSequence(){
        ++count;
    }

    //下面是把该类变成单例的代码
    private SequenceGenerator(){}
    private final static instance = newSequenceGenerator();
    public static SingleTon getInstance(){
        return instance;
    }

}
```

第四：

```
    public class MemoryDao
    {
    private HashMap map = new HashMap();

   publicvoid add(Student stu1){
            map.put(SequenceGenerator.getInstance().getSequence(),stu1);
    }
```

//把 MemoryDao 变成单例

}

Singleton 模式主要作用是保证在 Java 应用程序中，一个类 Class 只有一个实例存在。

一般 Singleton 模式通常有几种种形式:

第一种形式: 定义一个类，它的构造函数为 private 的，它有一个 static 的 private 的该类变量，在类初始化时实例话，通过一个 public 的 getInstance 方法获取对它的引用,继而调用其中的方法。

```
public class Singleton {
private Singleton(){}
    //在自己内部定义自己一个实例，是不是很奇怪？
    //注意这是 private 只供内部调用
    private staticSingleton instance = new Singleton();
    //这里提供了一个供外部访问本 class 的静态方法，可以直接访问
    public staticSingleton getInstance() {
    return instance;
    }
  }
```

第二种形式:

```
public class Singleton {
private static Singleton instance = null;
public static synchronized Singleton getInstance() {
//这个方法比上面有所改进，不用每次都进行生成对象，只是第一次
//使用时生成实例，提高了效率！
if (instance==null)
```

instance＝new Singleton();

return instance;

}

}

其他形式:

定义一个类，它的构造函数为 private 的，所有方法为 static 的。

一般认为第一种形式要更加安全些

8、递归算法题 1

一个整数，大于0，不用循环和本地变量，按照 n，2n，4n，8n 的顺序递增，当值大于5000时，把值按照指定顺序输出来。

例：n=1237

则输出为：

1237，

2474，

4948，

9896，

9896，

4948，

2474，

1237，

提示：写程序时，先致谢按递增方式的代码，写好递增的以后，再增加考虑递减部分。

```
public static void doubleNum(int n)
{
    System.out.println(n);
    if(n<=5000)
        doubleNum(n*2);
    System.out.println(n);
}
```

Gaibaota(N) = Gaibaota(N-1) + n

9、递归算法题 2

第1个人10，第2个比第1个人大2岁，依次递推，请用递归方式计算出第8个人多大？

```
package cn.itcast;

import java.util.Date;

publicclass A1 {

    public static voidmain(String [] args)
    {
        System.out.println(computeAge(8));
    }

    public static int computeAge(intn)
    {
        if(n==1)return 10;
        returncomputeAge(n-1) + 2;
    }
}

    public static voidtoBinary(int n,StringBuffer result)
    {

        if(n/2 != 0)
            toBinary(n/2,result);
        result.append(n%2);
    }
```

10、排序都有哪几种方法？请列举。用 JAVA 实现一个快速排序。

本人只研究过冒泡排序、选择排序和快速排序，下面是快速排序的代码：

```
public class QuickSort {
```

```
/**
* 快速排序
* @param strDate
* @param left
* @param right
*/
public void quickSort(String[] strDate,int left,int right){
String middle,tempDate;
int i,j;
i=left;
j=right;
middle=strDate[(i+j)/2];
do{
while(strDate[i].compareTo(middle)<0&& i<right)
i++; //找出左边比中间值大的数
while(strDate[j].compareTo(middle)>0&& j>left)
j--; //找出右边比中间值小的数
if(i<=j){ //将左边大的数和右边小的数进行替换
tempDate=strDate[i];
strDate[i]=strDate[j];
strDate[j]=tempDate;
i++;
j--;
}
}while(i<=j); //当两者交错时停止

if(i<right){
quickSort(strDate,i,right);//从
}
if(j>left){
quickSort(strDate,left,j);
}
}
/**
  * @param args
```

```
*/
public static void main(String[] args){
String[] strVoid=newString[]{"11","66","22","0","55","22","0","32"};
QuickSort sort=new QuickSort();
sort.quickSort(strVoid,0,strVoid.length-1);
for(int i=0;i<strVoid.length;i++){
System.out.println(strVoid[i]+" ");
}
}


}
```

11、有数组 **a[n]**，用 **java** 代码将数组元素顺序颠倒

```
//用下面的也可以
//for(inti=0,int j=a.length-1;i<j;i++,j--)是否等效于 for(int i=0;i<a.length/2;i++)呢？

importjava.util.Arrays;

public classSwapDemo{

    public static void main(String[] args){
        int [] a = new int[]{
                        (int)(Math.random() *1000),
                        (int)(Math.random() * 1000),
                        (int)(Math.random() *1000),
                        (int)(Math.random() *1000),
                        (int)(Math.random() * 1000)
        };

        System.out.println(a);
        System.out.println(Arrays.toString(a));
        swap(a);
```

```
        System.out.println(Arrays.toString(a));
    }

    public static void swap(int a[]){
        int len = a.length;
        for(int i=0;i<len/2;i++){
            int tmp = a[i];
            a[i] = a[len-1-i];
            a[len-1-i] = tmp;
        }
    }
}
```

12．金额转换，阿拉伯数字的金额转换成中国传统的形式如：（￥**1011**）**－>**（一千零一拾一元整）输出。

去零的代码：

```
    returnsb.reverse().toString().replaceAll("零[拾佰仟]","零").replaceAll("零+万","万
").replaceAll("零+元","元").replaceAll("零+","零");

public class RenMingBi {

        /**
         * @param args add by zxx ,Nov 29, 2008
         */
        private static finalchar[] data = new char[]{
                          '零','壹','贰','叁','肆','伍','陆','柒','捌','玖'
                  };
        private static finalchar[] units = new char[]{
                  '元','拾','佰','仟','万','拾','佰','仟','亿'
        };
        public static voidmain(String[] args) {
```

```java
		// TODOAuto-generated method stub
		System.out.println(
				convert(135689123));
	}

	public static Stringconvert(int money)
	{
		StringBuffersbf = new StringBuffer();
		int unit = 0;
		while(money!=0)
		{
			sbf.insert(0,units[unit++]);
			intnumber = money%10;
			sbf.insert(0,data[number]);
			money/= 10;
		}

		returnsbf.toString();
	}
}
```

三. **html&JavaScript&ajax** 部分

1. 判断第二个日期比第一个日期大

如何用脚本判断用户输入的的字符串是下面的时间格式2004-11-21必须要保证用户的输入是此格式，并且是时间，比如说月份不大于12等等，另外我需要用户输入两个，并且后一个要比前一个晚，只允许用 JAVASCRIPT，请详细帮助作答，，

//这里可用正则表达式判断提前判断一下格式，然后按下提取各时间字段内容

```
<script type="text/javascript">
  window.onload =function()
  {
     //这么写是为了实现 js 代码与 html 代码的分离，当我修改 js 时，不能影响 html 代码。
```

```
document.getElementById("frm1").onsubmit=
        function(){
                vard1 = this.d1.value;
                vard2 = this.d2.value;
                if(!verifyDate(d1)) {alert("第一个日期格式不对");return false;}
                if(!verifyDate(d2)) {alert("第二个日期格式不对");return false;}
                if(!compareDate(d1,d2)){alert("第二个日期比第一日期小");return
false;}
        };
}

functioncompareDate(d1,d2)
{
        var arrayD1= d1.split("-");
        var date1 =new Date(arrayD1[0],arrayD1[1],arrayD1[2]);
        var arrayD2= d2.split("-");
        var date2 =new Date(arrayD2[0],arrayD2[1],arrayD2[2]);
        if(date1> date2) return false;
        return true;
}

functionverifyDate(d)
{
        vardatePattern = /^\d{4}-(0?[1-9]|1[0-2])-(0?[1-9]|[1-2]\d|3[0-1])$/;
        returndatePattern.test(d);
}
</script>

<form id="frm1" action="xxx.html">
<input type="text" name="d1" />
<input type="text" name="d2" />
```

```
<input type="submit"/>
</form>
```

2. 用 **table** 显示 **n** 条记录，每 **3** 行换一次颜色，即 **1**，**2**，**3** 用红色字体，**4**，**5**，**6** 用绿色字体，**7**，**8**，**9** 用红颜色字体。

```
<body>
<table id="tbl">
    <tr><td>1</td></tr>
    <tr><td>2</td></tr>
    <tr><td>3</td></tr>
    <tr><td>4</td></tr>
    <tr><td>5</td></tr>
    <tr><td>6</td></tr>
    <tr><td>7</td></tr>
    <tr><td>8</td></tr>
    <tr><td>9</td></tr>
    <tr><td>10</td></tr>
</table>
</body>
<script type="text/javascript">
    window.onload=function()
          {
                    var tbl =document.getElementById("tbl");
                    rows =tbl.getElementsByTagName("tr");
                    for(i=0;i<rows.length;i++)
                    {
                              var j= parseInt(i/3);
                              if(j%2==0)rows[i].style.backgroundColor="#f00";
                              else   rows[i].style.backgroundColor="#0f0";
                    }
          }
```

```
</script>
```

3、**HTML** 的 **form** 提交之前如何验证数值文本框的内容全部为数字**?**否则的话提示用户并终止提交**?**

```
<form onsubmit='return chkForm(this)'>
<input type="text" name="d1"/>
<input type="submit"/>
</form>
<script type="text/javascript" />
function chkForm(this)
	{
		var value = thist.d1.value;
		var len =value.length;
		for(vari=0;i<len;i++)
		{
			if(value.charAt(i)>"9"|| value.charAt(i)<"0")
			{
				alert("含有非数字字符");
				returnfalse;
			}
		}
		return true;
	}
</script>
```

4、请写出用于校验 **HTML** 文本框中输入的内容全部为数字的 **javascript** 代码

```
<input type="text" id="d1" onblur=" chkNumber(this)"/>
<script type="text/javascript" />
function chkNumber(eleText)

	{
```

```
                var value =eleText.value;
                var len =value.length;
                for(vari=0;i<len;i++)
                {
                        if(value.charAt(i)>"9"|| value.charAt(i)<"0")
                        {
                                alert("含有非数字字符");
                                eleText.focus();
                                break;
                        }
                }
        }
</script>
```

除了写完代码，还应该在网页上写出实验步骤和在代码中加入实现思路，让面试官一看就明白你的意图和检查你的结果。

5、说说你用过那些 ajax 技术和框架，说说它们的区别

四. Java web 部分

1、Tomcat 的优化经验

答:去掉对 web.xml 的监视，把 jsp 提前编辑成 Servlet。

有富余物理内存的情况，加大 tomcat 使用的 jvm 的内存

2、HTTP 请求的 GET 与 POST 方式的区别

答:servlet 有良好的生存期的定义，包括加载和实例化、初始化、处理请求以及服务结束。这个生存期由 javax.servlet.Servlet 接口的 init,service 和 destroy 方法表达。

3、解释一下什么是 servlet;

答:servlet 有良好的生存期的定义，包括加载和实例化、初始化、处理请求以及服务结束。这个生存期由 javax.servlet.Servlet 接口的 init,service 和 destroy 方法表达。

4、说一说 **Servlet** 的生命周期?

答:servlet 有良好的生存期的定义，包括加载和实例化、初始化、处理请求以及服务结束。这个生存期由 javax.servlet.Servlet 接口的 init,service 和 destroy 方法表达。

Servlet 被服务器实例化后，容器运行其 init 方法，请求到达时运行其 service 方法，service 方法自动派遣运行与请求对应的 doXXX 方法（doGet，doPost）等，当服务器决定将实例销毁的时候调用其 destroy 方法。

web 容器加载 servlet，生命周期开始。通过调用 servlet 的 init()方法进行 servlet 的初始化。通过调用 service()方法实现，根据请求的不同调用不同的 do***()方法。结束服务，web 容器调用 servlet 的 destroy()方法。

5、Servlet 的基本架构

```
public class ServletName extends HttpServlet {
public void doPost(HttpServletRequest request,HttpServletResponse response) throws
ServletException, IOException {
}
public void doGet(HttpServletRequest request,HttpServletResponse response) throws
ServletException, IOException {
}
}
```

6、SERVLET API 中 forward()与 redirect()的区别？

答:前者仅是容器中控制权的转向，在客户端浏览器地址栏中不会显示出转向后的地址；后者则是完全的跳转，浏览器将会得到跳转的地址，并重新发送请求链接。这样，从浏览器的地址栏中可以看到跳转后的链接地址。所以，前者更加高效，在前者可以满足需要时，尽量使用 forward()方法，并且，这样也有助于隐藏实际的链接。在有些情况下，比如，需要跳转到一个其它服务器上的资源，则必须使用
sendRedirect()方法。

7、什么情况下调用 doGet()和 doPost()？

Jsp 页面中的 FORM 标签里的 method 属性为 get 时调用 doGet()，为 post 时调用 doPost()。

8、Request 对象的主要方法：

setAttribute(String name,Object)：设置名字为 name 的 request 的参数值

getAttribute(String name)：返回由 name 指定的属性值

getAttributeNames()：返回 request 对象所有属性的名字集合，结果是一个枚举的实例

getCookies()：返回客户端的所有 Cookie 对象，结果是一个 Cookie 数组

getCharacterEncoding()：返回请求中的字符编码方式

getContentLength()：返回请求的 Body 的长度

getHeader(String name)：获得 HTTP 协议定义的文件头信息

getHeaders(String name)：返回指定名字的 request Header 的所有值，结果是一个枚举的实例

getHeaderNames()：返回所以 request Header 的名字，结果是一个枚举的实例

getInputStream()：返回请求的输入流，用于获得请求中的数据

getMethod()：获得客户端向服务器端传送数据的方法

getParameter(String name)：获得客户端传送给服务器端的有 name 指定的参数值

getParameterNames()：获得客户端传送给服务器端的所有参数的名字，结果是一个枚举的实例

getParametervalues(String name)：获得有 name 指定的参数的所有值

getProtocol()：获取客户端向服务器端传送数据所依据的协议名称

getQueryString()：获得查询字符串

getRequestURI()：获取发出请求字符串的客户端地址

getRemoteAddr()：获取客户端的 IP 地址

getRemoteHost()：获取客户端的名字

getSession([Boolean create])：返回和请求相关 Session

getServerName()：获取服务器的名字

getServletPath()：获取客户端所请求的脚本文件的路径

getServerPort()：获取服务器的端口号

removeAttribute(String name)：删除请求中的一个属性

9、forward 和 redirect 的区别

forward 是服务器请求资源，服务器直接访问目标地址的 URL，把那个 URL 的响应内容读取过来，然后把这些内容再发给浏览器，浏览器根本不知道服务器发送的内容是从哪儿来的，所以它的地址栏中还是原来的地址。

redirect 就是服务端根据逻辑,发送一个状态码,告诉浏览器重新去请求那个地址，一般来说浏览器会用刚才请求的所有参数重新请求，所以 session,request 参数都可以获取。

10、request.getAttribute()和 request.getParameter()有何区别?

getParameter 得到的都是 String 类型的。或者是 http://a.jsp?id=123 中的 123，或者是某个表单提交过去的数据。
getAttribute 则可以是对象。
getParameter()是获取 POST/GET 传递的参数值；
getAttribute()是获取对象容器中的数据值；
getParameter：用于客户端重定向时，即点击了链接或提交按扭时传值用，即用于在用表单或 url 重定向传值时接收数据用。
getAttribute：用于服务器端重定向时，即在 sevlet 中使用了 forward 函数,或 struts 中使用了 mapping.findForward。getAttribute 只能收到程序用 setAttribute 传过来的值。
getParameter()是获取 POST/GET 传递的参数值；
getAttribute()是获取 SESSION 的值；
另外，可以用 setAttribute,getAttribute 发送接收对象.而 getParameter 显然只能传字符串。setAttribute 是应用服务器把这个对象放在该页面所对应的一块内存中去，当你的页面服务器重定向到另一个页面时，应用服务器会把这块内存拷贝另一个页面所对应的内存中。这样 getAttribute 就能取得你所设下的值，当然这种方法可以传对象。session 也一样，只是对象在内存中的生命周期不一样而已。getParameter 只是应用服务器在分析你送上来的 request 页面的文本时，取得你设在表单或 url 重定向时的值。
getParameter 返回的是 String, 用于读取提交的表单中的值;
getAttribute 返回的是 Object，需进行转换,可用 setAttribute 设置成任意对象，使用很灵活，可随时用；

11. jsp 有哪些内置对象?作用分别是什么?分别有什么方法？

答:JSP 共有以下9个内置的对象：

request 用户端请求，此请求会包含来自 GET/POST 请求的参数

response 网页传回用户端的回应

pageContext 网页的属性是在这里管理

session 与请求有关的会话期

application servlet 正在执行的内容

out 用来传送回应的输出

config servlet 的构架部件

page JSP 网页本身

exception 针对错误网页，未捕捉的例外

request 表示 HttpServletRequest 对象。它包含了有关浏览器请求的信息，并且提供了几个用于获取 cookie, header,和 session 数据的有用的方法。

response 表示 HttpServletResponse 对象，并提供了几个用于设置送回浏览器的响应的方法（如 cookies,头信息等）

out 对象是 javax.jsp.JspWriter 的一个实例，并提供了几个方法使你能用于向浏览器回送输出结果。

pageContext 表示一个 javax.servlet.jsp.PageContext 对象。它是用于方便存取各种范围的名字空间、servlet 相关的对象的 API，并且包装了通用的 servlet 相关功能的方法。

session 表示一个请求的 javax.servlet.http.HttpSession 对象。Session 可以存贮用户的状态信息

applicaton 表示一个 javax.servle.ServletContext 对象。这有助于查找有关 servlet 引擎和 servlet 环境的信息

config 表示一个 javax.servlet.ServletConfig 对象。该对象用于存取 servlet 实例的初始化参数。

page 表示从该页面产生的一个 servlet 实例

12. jsp 有哪些动作?作用分别是什么?

（这个问题似乎不重要，不明白为何有此题）

答:JSP 共有以下6种基本动作

jsp:include：在页面被请求的时候引入一个文件。

jsp:useBean：寻找或者实例化一个 JavaBean。

jsp:setProperty：设置 JavaBean 的属性。

jsp:getProperty：输出某个 JavaBean 的属性。

jsp:forward：把请求转到一个新的页面。

jsp:plugin：根据浏览器类型为 Java 插件生成 OBJECT 或 EMBED 标记

13、JSP 的常用指令

isErrorPage(是否能使用 Exception 对象)，isELIgnored(是否忽略表达式)

14. JSP 中动态 INCLUDE 与静态 INCLUDE 的区别？

答：动态 INCLUDE 用 jsp:include 动作实现

<jsp:include page=included.jsp flush=true />它总是会检查所含文件中的变化，适合用于包含动态页面，并且可以带参数 静态 INCLUDE 用 include 伪码实现,定不会检查所含文件的变化，适用于包含静态页面 <%@include file=included.htm %>

15、两种跳转方式分别是什么?有什么区别?

（下面的回答严重错误，应该是想问 forward 和 sendRedirect 的区别，毕竟出题的人不是专业搞文字艺术的人，可能表达能力并不见得很强，用词不一定精准，加之其自身的技术面也可能存在一些问题，不一定真正将他的意思表达清楚了，严格意思上来讲，一些题目可能根本就无人能答，所以，答题时要掌握主动，只要把自己知道的表达清楚就够了，而不要去推敲原始题目的具体含义是什么，不要一味想着是在答题）

答：有两种，分别为：

<jsp:include page=included.jsp flush=true>

<jsp:forward page= nextpage.jsp/>

前者页面不会转向 include 所指的页面，只是显示该页的结果，主页面还是原来的页面。执行完后还会回来，相当于函数调用。并且可以带参数.后者完全转向新页面，不会再回来。相当于 go to 语句。

16、页面间对象传递的方法

request，session，application，cookie 等

17、JSP 和 Servlet 有哪些相同点和不同点，他们之间的联系是什么？

JSP 是 Servlet 技术的扩展，本质上是 Servlet 的简易方式，更强调应用的外表表达。JSP 编译后是"类 servlet"。Servlet 和 JSP 最主要的不同点在于，Servlet 的应用逻辑是在 Java 文件中，并且完全从表示层中的 HTML 里分离开来。而 JSP 的情况是 Java 和 HTML 可以组合成一个扩展名为.jsp 的文件。JSP 侧重于视图，Servlet 主要用于控制逻辑。

18、MVC 的各个部分都有那些技术来实现?如何实现?

答:MVC 是 Model－View－Controller 的简写。Model 代表的是应用的业务逻辑（通过 JavaBean，EJB 组件实现），View 是应用的表示面（由 JSP 页面产生），Controller 是提供应用的处理过程控制（一般是一个 Servlet），通过这种设计模型把应用逻辑，处理过程和显示逻辑分成不同的组件实现。这些组件可以进行交互和重用。

19、我们在 web 应用开发过程中经常遇到输出某种编码的字符，如 iso8859-1 等，如何输出一个某种编码的字符串？

```
Public String translate(String str) {
  String tempStr ="";
```

```
    try {
      tempStr = newString(str.getBytes("ISO-8859-1"), "GBK");
      tempStr =tempStr.trim();
    }
    catch (Exception e) {
     System.err.println(e.getMessage());
    }
    return tempStr;
  }
```

20、现在输入 **n** 个数字，以逗号，分开；然后可选择升或者降序排序；按提交键就在另一页面显示按什么排序，结果为，提供 **reset**

五.数据库部分

1、用两种方式根据部门号从高到低，工资从低到高列出每个员工的信息。

employee:

eid,ename,salary,deptid;

```
select * from employeeorder by deptid desc,salary
```

2、列出各个部门中工资高于本部门的平均工资的员工数和部门号，并按部门号排序

创建表：

```
mysql> createtable employee921(id int primary key auto_increment,name
varchar(5
0),salary bigint,deptid int);
```

插入实验数据：

```
mysql> insert into employee921values(null,'zs',1000,1),(null,'ls',1100,1),(null
```

,'ww',1100,1),(null,'zl',900,1) ,(null,'zl',1000,2), (null,'zl',900,2),(null,'z
l',1000,2) , (null,'zl',1100,2);

编写 sql 语句：

（）select avg(salary) from employee921 group by deptid;

（）mysql>
selectemployee921.id,employee921.name,employee921.salary,employee921.dep
tid tid from employee921where salary > (select avg(salary) from employee921 where deptid = tid);

效率低的一个语句，仅供学习参考使用（在 group by 之后不能使用 where，只能使用 having，在 group by 之前可以使用 where，即表示对过滤后的结果分组）：

mysql>
selectemployee921.id,employee921.name,employee921.salary,employee921.dep
tid tid from employee921where salary > (select avg(salary) from employee921 group by deptid havingdeptid = tid);

```
（）select count(*) ,tid
        from (

selectemployee921.id,employee921.name,employee921.salary,employee921.deptid tid
                from        employee921
                where salary>
                        (selectavg(salary) from employee921 where deptid = tid)
        ) as t
        group by tid ;
```

另外一种方式：关联查询

```
select a.ename,a.salary,a.deptid
 from emp a,
    (selectdeptd,avg(salary) avgsal from emp group by deptid ) b
 where a.deptid=b.deptidand a.salary>b.avgsal;
```

3、存储过程与触发器必须讲，经常被面试到**?**

```
create procedure insert_Student (_name varchar(50),_age int ,out_id int)
begin
	insert into studentvalue(null,_name,_age);
	select max(stuId)into _id from student;
end;

call insert_Student('wfz',23,@id);
select @id;

mysql> create trigger update_Student BEFORE update on studentFOR EACH ROW
-> select * from student;
```

触发器不允许返回结果

```
create trigger update_StudentBEFORE update on student FOR EACH ROW
insert into   student value(null,'zxx',28);
```

mysql 的触发器目前不能对当前表进行操作

```
create trigger update_StudentBEFORE update on student FOR EACH ROW
delete from articles   where id=8;
```

这个例子不是很好，最好是用删除一个用户时，顺带删除该用户的所有帖子

这里要注意使用 OLD.id

触发器用处还是很多的，比如校内网、开心网、Facebook，你发一个日志，自动通知好友，其实就是在增加日志时做一个后触发，再向通知表中写入条目。因为触发器效率高。而 UCH 没有用触发器，效率和数据处理能力都很低。

存储过程的实验步骤：

```
mysql> delimiter |
mysql> create procedure insertArticle_Procedure (pTitle varchar(50),pBid int,out
 pId int)
    -> begin
```

```
    -> insert into article1value(null,pTitle,pBid);
    -> select max(id) into pId fromarticle1;
    -> end;
    -> |
Query OK, 0 rows affected (0.05sec)

mysql> callinsertArticle_Procedure('传智播客',1,@pid);
    -> |
Query OK, 0 rows affected (0.00sec)

mysql> delimiter ;
mysql> select @pid;
+------+
| @pid |
+------+
| 3    |
+------+
1 row in set (0.00 sec)

mysql> select * fromarticle1;
+----+-------------+------+
| id | title       | bid |
+----+-------------+------+
| 1  | test        | 1    |
| 2  | chuanzhiboke | 1    |
| 3  | 传智播客     | 1    |
+----+-------------+------+
3 rows in set (0.00 sec)
```

触发器的实验步骤：

create table board1(id intprimary key auto_increment,name varchar(50),ar

```
ticleCount int);

create table article1(id intprimary key auto_increment,title varchar(50)
,bid int referencesboard1(id));

delimiter |

create triggerinsertArticle_Trigger after insert on article1 for each ro
w begin
    -> update board1 setarticleCount=articleCount+1 where id= NEW.bid;
    -> end;
    -> |

delimiter ;

insert into board1 value(null,'test',0);

insert into article1value(null,'test',1);
```

还有，每插入一个帖子，都希望将版面表中的最后发帖时间，帖子总数字段进行同步更新，用触发器做效率就很高。下次课设计这样一个案例，写触发器时，对于最后发帖时间可能需要用 declare 方式声明一个变量，或者是用 NEW.posttime 来生成。

4、数据库三范式是什么？

第一范式（1NF）：字段具有原子性,不可再分。所有关系型数据库系统都满足第一范式）

数据库表中的字段都是单一属性的，不可再分。例如，姓名字段，其中的姓和名必须作为一个整体，无法区分哪部分是姓，哪部分是名，如果要区分出姓和名，必须设计成两个独立的字段。

第二范式（2NF）：

第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。

要求数据库表中的每个实例或行必须可以被惟一地区分。通常需要为表加上一个列，以存储各个实例的惟一标识。这个惟一属性列被称为主关键字或主键。

第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。简而言之，第二范式就是非主属性非部分依赖于主关键字。

第三范式的要求如下：

满足第三范式（3NF）必须先满足第二范式（2NF）。简而言之，第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。

所以第三范式具有如下特征：

1，每一列只有一个值

2，每一行都能区分。

3，每一个表都不包含其他表已经包含的非主关键字信息。

例如，帖子表中只能出现发帖人的 id，而不能出现发帖人的 id，还同时出现发帖人姓名，否则，只要出现同一发帖人 id 的所有记录，它们中的姓名部分都必须严格保持一致，这就是数据冗余。

5、说出一些数据库优化方面的经验?

用 PreparedStatement 一般来说比 Statement 性能高：一个 sql 发给服务器去执行，涉及步骤：语法检查、语义分析，编译，缓存

“inert into user values(1,1,1)”-à 二进制

“inert into user values(2,2,2)”-à 二进制

“inert into user values(?,?,?)”-à 二进制

有外键约束会影响插入和删除性能，如果程序能够保证数据的完整性，那在设计数据库时就去掉外键。（比喻：就好比免检产品，就是为了提高效率，充分相信产品的制造商）

（对于 hibernate 来说，就应该有一个变化：empleyee->Deptment 对象，现在设计时就成了 employeeàdeptid）

看 mysql 帮助文档子查询章节的最后部分，例如，根据扫描的原理，下面的子查询语句要比第二条关联查询的效率高：

1. select e.name,e.salarywhere e.managerid=(select id from employee where name='zxx');

2. select e.name,e.salary,m.name,m.salary fromemployees e,employees m where e.managerid = m.id andm.name='zxx';

表中允许适当冗余，譬如，主题帖的回复数量和最后回复时间等

将姓名和密码单独从用户表中独立出来。这可以是非常好的一对一的案例哟！

sql 语句全部大写，特别是列名和表名都大写。特别是 sql 命令的缓存功能，更加需要统一大小写，sql 语句 à 发给 oracle 服务器 à 语法检查和编译成为内部指令 à 缓存和执行指令。根据缓存的特点，不要拼凑条件，而是用?和 PreparedStatment

还有索引对查询性能的改进也是值得关注的。

备注：下面是关于性能的讨论举例

4航班 3个城市

m*n

select * from flight,city where flight.startcityid=city.cityidand city.name='beijing';

m + n

select * from flight where startcityid = (select cityid fromcity where cityname='beijing');

select flight.id,'beijing',flight.flightTime from flight wherestartcityid = (select cityid from city where cityname='beijing')

6、union 和 union all 有什么不同?

假设我们有一个表 Student，包括以下字段与数据：

```
drop table student;
create table student
(
id int primary key,
name nvarchar2(50) not null,
score number not null
);
insert into student values(1,'Aaron',78);
insert into student values(2,'Bill',76);
insert into student values(3,'Cindy',89);
insert into student values(4,'Damon',90);
insert into student values(5,'Ella',73);
insert into student values(6,'Frado',61);
insert into student values(7,'Gill',99);
insert into student values(8,'Hellen',56);
insert into student values(9,'Ivan',93);
insert into student values(10,'Jay',90);
commit;
```

Union 和 Union All 的区别。

```
select *
from student
where id < 4
union
select *
from student
where id > 2 and id < 6
```

结果将是

```
1    Aaron    78
2    Bill    76
```

3 Cindy 89

4 Damon 90

5 Ella 73

如果换成 Union All 连接两个结果集，则返回结果是：

1 Aaron 78

2 Bill 76

3 Cindy 89

3 Cindy 89

4 Damon 90

5 Ella 73

可以看到，Union 和 Union All 的区别之一在于对重复结果的处理。

UNION 在进行表链接后会筛选掉重复的记录，所以在表链接后会对所产生的结果集进行排序运算，删除重复的记录再返回结果。实际大部分应用中是不会产生重复的记录，最常见的是过程表与历史表 UNION。如：

select * from gc_dfys

union

select * from ls_jg_dfys

这个 SQL 在运行时先取出两个表的结果，再用排序空间进行排序删除重复的记录，最后返回结果集，如果表数据量大的话可能会导致用磁盘进行排序。

而 UNION ALL 只是简单的将两个结果合并后就返回。这样，如果返回的两个结果集中有重复的数据，那么返回的结果集就会包含重复的数据了。

从效率上说，UNION ALL 要比 UNION 快很多，所以，如果可以确认合并的两个结果集中不包含重复的数据的话，那么就使用 UNION ALL，

7.分页语句

取出 **sql** 表中第**31**到**40**的记录（以自动增长 **ID** 为主键）

sql server 方案1：

selecttop 10 * from t where id not in (select top 30 id from t order by id) orde byid

sql server 方案2：

selecttop 10 * from t where id in (select top 40 id from t order by id) order by iddesc

mysql 方案：select * from t order by idlimit 30,10

oracle 方案：select * from (select rownum r,* from t where r<=40) wherer>30

--------------------待整理进去的内容-------------------------------------

pageSize=20;

pageNo = 5;

1.分页技术1（直接利用 sql 语句进行分页，效率最高和最推荐的）

```
mysql:sql = "select * from articles limit " +(pageNo-1)*pageSize + "," + pageSize;
oracle: sql = "select * from " +
                                                        "(selectrownum r,* from " +
                                                                "(select* from
articles order by postime desc)" +
                                                        "whererownum<= " +
pageNo*pageSize +") tmp " +
                                                "wherer>" +
(pageNo-1)*pageSize;
```

注释：第7行保证 rownum 的顺序是确定的，因为 oracle 的索引会造成 rownum 返回不同的值

简洋提示：没有 order by 时，rownum 按顺序输出，一旦有了 order by，rownum 不按顺序输出了，这说明 rownum 是排序前的编号。如果对 order by 从句中的字段建立了索引，那么，rownum 也是按顺序输出的，因为这时候生成原始的查询结果集时会参照索引表的顺序来构建。

```
sqlserver:sql = "select top 10 * from id not id(select top" + (pageNo-1)*pageSize + "id from articles)"
```

```
DataSource ds = new InitialContext().lookup(jndiurl);
Connection cn = ds.getConnection();
//"select * from user where id=?"    --->binary directive
PreparedStatement pstmt = cn.prepareSatement(sql);
ResultSet rs = pstmt.executeQuery()
```

```
while(rs.next())
{
        out.println(rs.getString(1));
}
```

2.不可滚动的游标

```
pageSize=20;
pageNo = 5;
cn = null
stmt = null;
rs = null;
try
{
sqlserver:sql = "select * from articles";

DataSource ds = new InitialContext().lookup(jndiurl);
Connection cn = ds.getConnection();
//"select * from user where id=?"   --->binary directive
PreparedStatement pstmt = cn.prepareSatement(sql);
ResultSet rs = pstmt.executeQuery()
for(int j=0;j<(pageNo-1)*pageSize;j++)
{
        rs.next();
}

int i=0;

while(rs.next() && i<10)
{
        i++;
        out.println(rs.getString(1));
```

```
}
}
cacth(){}
finnaly
{
        if(rs!=null)try{rs.close();}catch(Exceptione){}
        if(stm.........
        if(cn............
}
```

3.可滚动的游标

```
pageSize=20;
pageNo = 5;
cn = null
stmt = null;
rs = null;
try
{
sqlserver:sql = "select * from articles";

DataSource ds = new InitialContext().lookup(jndiurl);
Connection cn = ds.getConnection();
//"select * from user where id=?"   --->binary directive
PreparedStatement pstmt =
cn.prepareSatement(sql,ResultSet.TYPE_SCROLL_INSENSITIVE,...);
//根据上面这行代码的异常 SQLFeatureNotSupportedException，就可判断驱动是否支持可
滚动游标

ResultSet rs = pstmt.executeQuery()
rs.absolute((pageNo-1)*pageSize)
int i=0;
```

```
while(rs.next() && i<10)
{
	i++;
	out.println(rs.getString(1));
}
}
cacth(){}
finnaly
{
	if(rs!=null)try{rs.close();}catch(Exceptione){}
	if(stm.........
	if(cn............
}
```

8.用一条 **SQL** 语句查询出每门课都大于 **80** 分的学生姓名

name	kecheng	fenshu
张三	语文	81
张三	数学	75
李四	语文	76
李四	数学	90
王五	语文	81
王五	数学	100
王五	英语	90

准备数据的 sql 代码：

```
create table score(id int primary key auto_increment,namevarchar(20),subject
varchar(20),score int);
insert into score values
(null,'张三','语文',81),
(null,'张三','数学',75),
(null,'李四','语文',76),
(null,'李四','数学',90),
(null,'王五','语文',81),
```

(null,'王五','数学',100),

(null,'王五 ','英语',90);

提示：当百思不得其解时，请理想思维，把小变成大做，把大变成小做，

答案：

A: select distinct name from score where name not in (selectdistinct name from score where score<=80)

B:select distince name t1 from score where 80< all (selectscore from score where name=t1);

9.所有部门之间的比赛组合

一个叫 department 的表，里面只有一个字段 name,一共有4条纪录，分别是 a,b,c,d,对应四个球对，现在四个球对进行比赛，用一条 sql 语句显示所有可能的比赛组合.

答：select a.name,b.name

from team a, team b

where a.name < b.name

10.每个月份的发生额都比 101 科目多的科目

请用 SQL 语句实现：从 TestDB 数据表中查询出所有月份的发生额都比101科目相应月份的发生额高的科目。请注意：TestDB 中有很多科目，都有1－12月份的发生额。

AccID：科目代码，Occmonth：发生额月份，DebitOccur：发生额。

数据库名：JcyAudit，数据集：Select * from TestDB

准备数据的 sql 代码：

drop table if exists TestDB;

create table TestDB(id int primary key auto_increment,AccIDvarchar(20), Occmonth date, DebitOccur bigint);

insert into TestDB values

(null,'101','1988-1-1',100),

(null,'101','1988-2-1',110),

(null,'101','1988-3-1',120),

(null,'101','1988-4-1',100),

(null,'101','1988-5-1',100),

(null,'101','1988-6-1',100),

(null,'101','1988-7-1',100),

(null,'101','1988-8-1',100);

--复制上面的数据，故意把第一个月份的发生额数字改小一点

insert into TestDB values

(null,'102','1988-1-1',90),

(null,'102','1988-2-1',110),

(null,'102','1988-3-1',120),

(null,'102','1988-4-1',100),

(null,'102','1988-5-1',100),

(null,'102','1988-6-1',100),

(null,'102','1988-7-1',100),

(null,'102','1988-8-1',100);

--复制最上面的数据，故意把所有发生额数字改大一点

insert into TestDB values

(null,'103','1988-1-1',150),

(null,'103','1988-2-1',160),

(null,'103','1988-3-1',180),

(null,'103','1988-4-1',120),

(null,'103','1988-5-1',120),

(null,'103','1988-6-1',120),

(null,'103','1988-7-1',120),

(null,'103','1988-8-1',120);

--复制最上面的数据，故意把所有发生额数字改大一点

insert into TestDB values

(null,'104','1988-1-1',130),

(null,'104','1988-2-1',130),

(null,'104','1988-3-1',140),

(null,'104','1988-4-1',150),

(null,'104','1988-5-1',160),

(null,'104','1988-6-1',170),

(null,'104','1988-7-1',180),

(null,'104','1988-8-1',140);

--复制最上面的数据，故意把第二个月份的发生额数字改小一点

insert into TestDB values

(null,'105','1988-1-1',100),

(null,'105','1988-2-1',80),

(null,'105','1988-3-1',120),

(null,'105','1988-4-1',100),

(null,'105','1988-5-1',100),

(null,'105','1988-6-1',100),

(null,'105','1988-7-1',100),

(null,'105','1988-8-1',100);

答案：

```
select distinct AccID from TestDB
where AccID not in
        (selectTestDB.AccIDfrom TestDB,
                (select * from TestDB where AccID='101') asdb101
        whereTestDB.Occmonth=db101.Occmonth and
TestDB.DebitOccur<=db101.DebitOccur
        );
```

11.统计每年每月的信息

year monthamount

1991 1 1.1

1991 2 1.2

1991 3 1.3

1991 4 1.4

1992 1 2.1

1992　2　　2.2

1992　3　　2.3

1992　4　　2.4

查成这样一个结果

year m1　m2　m3　m4

1991 1.1 1.2 1.3 1.4

1992 2.1 2.2 2.3 2.4

提示：这个与工资条非常类似，与学生的科目成绩也很相似。

准备 sql 语句：

```
drop table if existssales;
create table sales(idint auto_increment primary key,year varchar(10), month varchar(10),
amountfloat(2,1));
insert into salesvalues
(null,'1991','1',1.1),
(null,'1991','2',1.2),
(null,'1991','3',1.3),
(null,'1991','4',1.4),
(null,'1992','1',2.1),
(null,'1992','2',2.2),
(null,'1992','3',2.3),
(null,'1992','4',2.4);
```

答案一、

```
select sales.year ,
(select t.amount fromsales t where t.month='1' and t.year= sales.year) '1',
(select t.amount fromsales t where t.month='1' and t.year= sales.year) '2',
(select t.amount fromsales t where t.month='1' and t.year= sales.year) '3',
(select t.amount fromsales t where t.month='1' and t.year= sales.year) as '4'
from sales   group by year;
```

12.显示文章标题，发帖人、最后回复时间

表：id,title,postuser,postdate,parentid

准备 sql 语句：

drop table if exists articles;

create table articles(id int auto_increment primary key,titlevarchar(50), postuser varchar(10), postdate datetime,parentid int referencesarticles(id));

insert into articles values

(null,'第一条','张三','1998-10-10 12:32:32',null),

(null,'第二条','张三','1998-10-10 12:34:32',null),

(null,'第一条回复1','李四','1998-10-10 12:35:32',1),

(null,'第二条回复1','李四','1998-10-10 12:36:32',2),

(null,'第一条回复2','王五','1998-10-10 12:37:32',1),

(null,'第一条回复3','李四','1998-10-10 12:38:32',1),

(null,'第二条回复2','李四','1998-10-10 12:39:32',2),

(null,'第一条回复4','王五','1998-10-10 12:39:40',1);

答案：

select a.title,a.postuser,

(selectmax(postdate) from articles where parentid=a.id) reply

from articles a where a.parentid is null;

注释：子查询可以用在选择列中，也可用于 where 的比较条件中，还可以用于 from 从句中。

13.删除除了 id 号不同,其他都相同的学生冗余信息

2.学生表如下:

id 号	学号	姓名	课程编号	课程名称	分数
1	2005001	张三	0001	数学	69
2	2005002	李四	0001	数学	89
3	2005001	张三	0001	数学	69

A: delete from tablename where id 号 not in(select min(id 号) from tablename group by 学号,姓名,课程编号,课程名称,分数)

实验：

create table student2(id int auto_increment primary key,codevarchar(20),name varchar(20));

insert into student2 values(null,'2005001','张三'),(null,'2005002','李四'),(null,'2005001','张三');

//如下语句，mysql 报告错误，可能删除依赖后面统计语句，而删除又导致统计语句结果不一致。

delete from student2 where id not in(select min(id) fromstudent2 group by name);

//但是，如下语句没有问题：

select * from student2where id not in(select min(id) from student2 group by name);

//于是，我想先把分组的结果做成虚表，然后从虚表中选出结果，最后再将结果作为删除的条件数据。

delete from student2 where id not in(select mid from (selectmin(id) mid

from student2 group by name) as t);

或者：

delete from student2 where id not in(select min(id) from (select* from s

tudent2) as t group by t.name);

14.航空网的几个航班查询题：

表结构如下：

flight{flightID,StartCityID ,endCityID,StartTime}

city{cityID, CityName)

实验环境：

create table city(cityID int auto_increment primary key,cityNamevarchar(20));

create table flight (flightID int auto_increment primary key,

StartCityID intreferences city(cityID),

endCityID int references city(cityID),

StartTimetimestamp);

//航班本来应该没有日期部分才好，但是下面的题目当中涉及到了日期

insert into city values(null,'北京'),(null,'上海'),(null,'广州');

```
insert into flight values
        (null,1,2,'9:37:23'),(null,1,3,'9:37:23'),(null,1,2,'10:37:23'),(null,2,3,'10:37:23');
```

1、查询起飞城市是北京的所有航班，按到达城市的名字排序

参与运算的列是我起码能够显示出来的那些列，但最终我不一定把它们显示出来。各个表组合出来的中间结果字段中必须包含所有运算的字段。

```
  select   * from flight f,city c
        where f.endcityid =c.cityid and startcityid =
        (select c1.cityidfrom city c1 where c1.cityname = "北京")
        order by c.citynameasc;
```

```
mysql> select flight.flightid,'北京' startcity, e.cityname from flight,city e wh
ere flight.endcityid=e.cityid and flight.startcityid=(selectcityid from city wh
ere cityname='北京');
```

```
mysql> select flight.flightid,s.cityname,e.cityname fromflight,city s,city e wh
ere flight.startcityid=s.cityid and s.cityname='北京' andflight.endCityId=e.cit
yID order by e.cityName desc;
```

2、查询北京到上海的所有航班纪录（起飞城市，到达城市，起飞时间，航班号）

```
select c1.CityName,c2.CityName,f.StartTime,f.flightID
from city c1,city c2,flight f
where f.StartCityID=c1.cityID
and f.endCityID=c2.cityID
and c1.cityName='北京'
and c2.cityName='上海'
```

3、查询具体某一天（2005-5-8）的北京到上海的的航班次数

select count(*) from

(select c1.CityName,c2.CityName,f.StartTime,f.flightID

from city c1,city c2,flight f

where f.StartCityID=c1.cityID

and f.endCityID=c2.cityID

and c1.cityName='北京'

and c2.cityName='上海'

and 查帮助获得的某个日期处理函数(startTime) like '2005-5-8%'

mysql 中提取日期部分进行比较的示例代码如下：

select * from flight wheredate_format(starttime,'%Y-%m-%d')='1998-01-02'

15.查出比经理薪水还高的员工信息：

Drop table if not exists employees;

create table employees(id int primary key auto_increment,namevarchar(50)

,salary int,managerid intreferences employees(id));

insert into employees values (null,' lhm',10000,null), (null,'zxx',15000,1

),(null,'flx',9000,1),(null,'tg',10000,2),(null,'wzg',10000,3);

Wzg 大于 flx,lhm 大于 zxx

解题思路：

根据 sql 语句的查询特点，是逐行进行运算，不可能两行同时参与运算。

涉及了员工薪水和经理薪水，所有，一行记录要同时包含两个薪水，所有想到要把这个表自关联组合一下。

首先要组合出一个包含有各个员工及该员工的经理信息的长记录，譬如，左半部分是员工，右半部分是经理。而迪卡尔积会组合出很多垃圾信息，先去除这些垃圾信息。

select e.* from employees e,employees m where e.managerid=m.idand e.sala

ry>m.salary;

16、求出小于 45 岁的各个老师所带的大于 12 岁的学生人数

数据库中有3个表 teacher 表，student 表，tea_stu 关系表。

teacher 表 teaID name age

student 表 stuID name age

teacher_student 表 teaID stuID

要求用一条 sql 查询出这样的结果

1.显示的字段要有老师 name, age 每个老师所带的学生人数

2 只列出老师 age 为40以下，学生 age 为12以上的记录

预备知识：

1.sql 语句是对每一条记录依次处理，条件为真则执行动作（select,insert,delete,update）

2.只要是迪卡尔积，就会产生“垃圾”信息，所以，只要迪卡尔积了，我们首先就要想到清除“垃圾”信息

实验准备：

```
drop table if exists tea_stu;

drop table if exists teacher;

drop table if exists student;

create table teacher(teaID int primarykey,name varchar(50),age int);

create table student(stuID int primarykey,name varchar(50),age int);

create table tea_stu(teaID int referencesteacher(teaID),stuID int references student(stuID));

insertinto teacher values(1,'zxx',45), (2,'lhm',25) , (3,'wzg',26) , (4,'tg',27);

insertinto student values(1,'wy',11), (2,'dh',25) , (3,'ysq',26) , (4,'mxc',27);

insertinto tea_stu values(1,1), (1,2), (1,3);

insertinto tea_stu values(2,2), (2,3), (2,4);

 insert into tea_stu values(3,3), (3,4), (3,1);

insertinto tea_stu values(4,4), (4,1), (4,2) , (4,3);
```

结果：2à3,3à2,4à3

解题思路：（真实面试答题时，也要写出每个分析步骤，如果纸张不够，就找别人要）

1要会统计分组信息，统计信息放在中间表中：

selectteaid,count(*) from tea_stu group by teaid;

2接着其实应该是筛除掉小于12岁的学生，然后再进行统计，中间表必须与 student 关联才能得到12岁以下学生和把该学生记录从中间表中剔除，代码是：

selecttea_stu.teaid,count(*) total from student,tea_stu

wherestudent.stuid=tea_stu.stuid and student.age>12 group by tea_stu.teaid

3.接着把上面的结果做成虚表与 teacher 进行关联，并筛除大于45的老师

selectteacher.teaid,teacher.name,total from teacher ,***(select tea_stu.tea id,count(*)total from student,tea_stu where student.stuid=tea_stu.stuid and stu dent.age>12group by tea_stu.teaid) as tea_stu2***where teacher.teaid=tea_stu2.tea idand teacher.age<45;

17.求出发帖最多的人：

```
select authorid,count(*) total from articles
group by authorid
having total=
(select max(total2) from(select count(*) total2 from articles group by authorid) as t);
```

```
select t.authorid,max(t.total) from
（select authorid,count(*) total from articles）as t
```

这条语句不行，因为 max 只有一列，不能与其他列混淆。

select authorid,count(*) total from articles

group by authorid having total=max(total)也不行。

18、一个用户表中有一个积分字段，假如数据库中有 **100** 多万个用户，若要在每年第一天凌晨将积分清零，你将考虑什么，你将想什么办法解决**?**

alter table drop column score;

alter table add colunm score int;

可能会很快，但是需要试验，试验不能拿真实的环境来操刀，并且要注意，

这样的操作时无法回滚的，在我的印象中，只有 inert update delete 等 DML 语句才能回滚，

对于 create table,drop table ,alter table 等 DDL 语句是不能回滚。

解决方案一，update user set score=0;

解决方案二，假设上面的代码要执行好长时间，超出我们的容忍范围，那我就 alter table user drop columnscore;alter table user add column score int。

下面代码实现每年的那个凌晨时刻进行清零。

```
Runnable runnable =
        new Runnable(){
                public voidrun(){
                        clearDb();
                        schedule(this,newDate(new Date().getYear()+1,0,0));
                        }
                        };


schedule(runnable,
        new Date(newDate().getYear()+1,0,1));
```

19、一个用户具有多个角色，请查询出该表中具有该用户的所有角色的其他用户。

```
select count(*) as num,tb.id
from
 tb,
 (select role from tbwhere id=xxx) as t1
where
 tb.role = t1.role andtb.id != t1.id
```

group by tb.id

having

    num = select count(role)from tb where id=xxx;

20. xxx 公司的 sql 面试

Table **EMPLOYEES** Structure:

EMPLOYEE_ID NUMBER Primary Key,

FIRST_NAME VARCHAR2(25),

LAST_NAME VARCHAR2(25),

Salary number(8,2),

HiredDate DATE,

Departmentid number(2)

Table**Departments** Structure:

Departmentid number(2) Primary Key,

DepartmentName VARCHAR2(25).

(2）基于上述 EMPLOYEES 表写出查询：写出雇用日期在今年的，或者工资在[1000,2000]之间的，或者员工姓名（last_name）以'Obama'打头的所有员工，列出这些员工的全部个人信息。（4分）

select* from employees

whereYear(hiredDate) = Year(date())

    or (salary between 1000 and 200)

    or left(last_name,3)='abc';

(3) 基于上述 EMPLOYEES 表写出查询：查出部门平均工资大于1800元的部门的所有员工，列出这些员工的全部个人信息。（4分）

mysql>select id,name,salary,deptid did from employee1 where (select avg(salary)

 from employee1 where deptid = did) > 1800;

(4) 基于上述 EMPLOYEES 表写出查询：查出个人工资高于其所在部门平均工资的员工，列出这些员工的全部个人信息及该员工工资高出部门平均工资百分比。（5分）

selectemployee1.*,(employee1.salary-t.avgSalary)*100/employee1.salary

fromemployee1,

(select deptid,avg(salary) avgSalary fromemployee1 group by deptid) as t

whereemployee1.deptid = t.deptid and employee1.salary>t.avgSalary;

21、注册 Jdbc 驱动程序的三种方式

22、用 JDBC 如何调用存储过程

代码如下：

```
package com.huawei.interview.lym;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Types;

publicclass JdbcTest {

    /**
     * @paramargs
     */
    public static voidmain(String[] args) {
        // TODO Auto-generated method stub
        Connection cn = null;
        CallableStatement cstmt =null;
        try {
            //这里最好不要这么干，因为驱动名写死在程序中了
            Class.forName("com.mysql.jdbc.Driver");
            //实际项目中，这里应用 DataSource 数据，如果用框架，
            //这个数据源不需要我们编码创建，我们只需 Datasource ds =context.lookup()
```

```java
            //cn = ds.getConnection();
            cn = DriverManager.getConnection("jdbc:mysql:///test","root","root");
            cstmt = cn.prepareCall("{callinsert_Student(?,?,?)}");
            cstmt.registerOutParameter(3,Types.INTEGER);
            cstmt.setString(1,"wangwu");
            cstmt.setInt(2, 25);
            cstmt.execute();
            //get 第几个，不同的数据库不一样，建议不写
            System.out.println(cstmt.getString(3));
        } catch (Exception e) {
            // TODO Auto-generated catchblock
            e.printStackTrace();
        }
        finally
        {

            /*try{cstmt.close();}catch(Exceptione){}
            try{cn.close();}catch(Exceptione){}*/
            try {
                if(cstmt !=null)
                    cstmt.close();
                if(cn !=null)
                    cn.close();
            } catch (SQLException e) {
                // TODO Auto-generatedcatch block
                e.printStackTrace();
            }
        }
    }
```

23、JDBC 中的 PreparedStatement 相比 Statement 的好处

答：一个 sql 命令发给服务器去执行的步骤为：语法检查，语义分析，编译成内部指令，缓存指令，执行指令等过程。

select * from student where id =3----缓存--àxxxxx 二进制命令

select * from student where id =3----直接取-àxxxxx 二进制命令

select * from student where id =4--- -à 会怎么干？

如果当初是 select * from student where id =?--- -à 又会怎么干？

上面说的是性能提高

可以防止 sql 注入。

24. 写一个用 jdbc 连接并访问 oracle 数据的程序代码

25、Class.forName 的作用?为什么要用?

答：按参数中指定的字符串形式的类名去搜索并加载相应的类，如果该类字节码已经被加载过，则返回代表该字节码的 Class 实例对象，否则，按类加载器的委托机制去搜索和加载该类，如果所有的类加载器都无法加载到该类，则抛出 ClassNotFoundException。加载完这个 Class 字节码后，接着就可以使用 Class 字节码的 newInstance 方法去创建该类的实例对象了。

有时候，我们程序中所有使用的具体类名在设计时（即开发时）无法确定，只有程序运行时才能确定，这时候就需要使用 Class.forName 去动态加载该类，这个类名通常是在配置文件中配置的，例如，spring 的 ioc 中每次依赖注入的具体类就是这样配置的，jdbc 的驱动类名通常也是通过配置文件来配置的，以便在产品交付使用后不用修改源程序就可以更换驱动类名。

26、大数据量下的分页解决方法。

答：最好的办法是利用 sql 语句进行分页，这样每次查询出的结果集中就只包含某页的数据内容。再 sql 语句无法实现分页的情况下，可以考虑对大的结果集通过游标定位方式来获取某页的数据。

sql 语句分页，不同的数据库下的分页方案各不一样，下面是主流的三种数据库的分页 sql：

sql server:

```
String sql =
"select top" + pageSize + " * from students where id not in" +

"(select top "+ pageSize * (pageNumber-1) + " id from students order by id)" +
```

"order by id";

mysql:

```
String sql =
"select * fromstudents order by id limit " + pageSize*(pageNumber-1) + ","
+pageSize;
```

oracle:

```
String sql =
 "select * from " +
 (select *,rownum rid from (select * fromstudents order by postime desc) where
rid<=" + pagesize*pagenumber +") as t" +
 "where t>" +pageSize*(pageNumber-1);
```

27、用 **JDBC** 查询学生成绩单**,**把主要代码写出来（考试概率极大）**.**

```
Connection cn = null;
PreparedStatement pstmt =null;
Resultset rs = null;
try
{
    Class.forname(driveClassName);
    cn =  DriverManager.getConnection(url,username,password);
    pstmt =cn.prepareStatement("select  score.* fromscore ,student " +
        "wherescore.stuId = student.id and student.name = ?");
    pstmt.setString(1,studentName);
    Resultset rs =pstmt.executeQuery();
    while(rs.next())
    {
        system.out.println(rs.getInt("subject")  +  "    " +
```

```
rs.getFloat("score") );
        }
}catch(Exception e){e.printStackTrace();}
finally
{
        if(rs != null) try{rs.close() }catch(exception e){}
        if(pstmt != null)try{pstmt.close()}catch(exception e){}
        if(cn != null) try{cn.close() }catch(exception e){}
}
```

28、这段代码有什么不足之处?

```
try {
Connection conn = ...;
Statement stmt = ...;
ResultSet rs =stmt.executeQuery("select * from table1");
while(rs.next()) {
}
} catch(Exception ex) {
}
```

答：没有 finally 语句来关闭各个对象，另外，使用 finally 之后，要把变量的定义放在 try 语句块的外面，以便在 try 语句块之外的 finally 块中仍可以访问这些变量。

29、说出数据连接池的工作机制是什么?

J2EE 服务器启动时会建立一定数量的池连接，并一直维持不少于此数目的池连接。客户端程序需要连接时，池驱动程序会返回一个未使用的池连接并将其表记为忙。如果当前没有空闲连接，池驱动程序就新建一定数量的连接，新建连接的数量有配置参数决定。当使用的池连接调用完成后，池驱动程序将此连接表记为空闲，其他调用就可以使用这个连接。

实现方式，返回的 Connection 是原始 Connection 的代理，代理 Connection 的 close 方法不是真正关连接，而是把它代理的 Connection 对象还回到连接池中。

30、为什么要用 ORM? 和 JDBC 有何不一样?

orm 是一种思想，就是把 object 转变成数据库中的记录，或者把数据库中的记录转变成 objecdt，我们可以用 jdbc 来实现这种思想，其实，如果我们的项目是严格按照 oop 方式编写的话，我们的 jdbc 程序不管是有意还是无意，就已经在实现 orm 的工作了。

现在有许多 orm 工具，它们底层调用 jdbc 来实现了 orm 工作，我们直接使用这些工具，就省去了直接使用 jdbc 的繁琐细节，提高了开发效率，现在用的较多的 orm 工具是 hibernate。也听说一些其他 orm 工具，如 toplink,ojb 等。

六. XML 部分

1、xml 有哪些解析技术?区别是什么?

答:有 DOM,SAX,STAX 等

DOM:处理大型文件时其性能下降的非常厉害。这个问题是由 DOM 的树结构所造成的，这种结构占用的内存较多，而且 DOM 必须在解析文件之前把整个文档装入内存,适合对 XML 的随机访问 SAX:不现于 DOM,SAX 是事件驱动型的 XML 解析方式。它顺序读取 XML 文件，不需要一次全部装载整个文件。当遇到像文件开头，文档结束，或者标签开头与标签结束时，它会触发一个事件，用户通过在其回调事件中写入处理代码来处理 XML 文件，适合对 XML 的顺序访问

STAX:Streaming API for XML (StAX)

讲解这些区别是不需要特别去比较，就像说传智播客与其他培训机构的区别时，我们只需说清楚传智播客有什么特点和优点就行了，这就已经间接回答了彼此的区别。

2、你在项目中用到了 xml 技术的哪些方面?如何实现的?

答:用到了数据存贮，信息配置两方面。在做数据交换平台时，将不能数据源的数据组装成 XML 文件，然后将 XML 文件压缩打包加密后通过网络传送给接收者，接收解密与解压缩后再同 XML 文件中还原相关信息进行处理。在做软件配置时，利用 XML 可以很方便的进行，软件的各种配置参数都存贮在 XML 文件中。

3、用 jdom 解析 xml 文件时如何解决中文问题?如何解析?

答:看如下代码,用编码方式加以解决

```
package test;
import java.io.*;
public class DOMTest
{
private String inFile = "c:\\people.xml"
```

```java
private String outFile = "c:\\people.xml"
public static void main(String args[])
{
new DOMTest();
}
public DOMTest()
{
try
{
javax.xml.parsers.DocumentBuilder builder =
javax.xml.parsers.DocumentBuilderFactory.newInstance().newDocumentBuilder();
org.w3c.dom.Document doc = builder.newDocument();
org.w3c.dom.Element root = doc.createElement("老师");
org.w3c.dom.Element wang = doc.createElement("王");
org.w3c.dom.Element liu = doc.createElement("刘");
wang.appendChild(doc.createTextNode("我是王老师"));
root.appendChild(wang);
doc.appendChild(root);
javax.xml.transform.Transformer transformer =
javax.xml.transform.TransformerFactory.newInstance().newTransformer();
transformer.setOutputProperty(javax.xml.transform.OutputKeys.ENCODING,"gb2312");
transformer.setOutputProperty(javax.xml.transform.OutputKeys.INDENT,"yes");
transformer.transform(newjavax.xml.transform.dom.DOMSource(doc),
new
javax.xml.transform.stream.StreamResult(outFile));
}
catch (Exception e)
{
System.out.println (e.getMessage());
}
}
}
```

4、编程用 JAVA 解析 XML 的方式.

答:用 SAX 方式解析 XML，XML 文件如下：

```xml
<?xml version=1.0 encoding=gb2312?>
<person>
<name>王小明</name>
<college>信息学院</college>
<telephone>6258113</telephone>
<notes>男,1955年生,博士，95年调入海南大学</notes>
</person>
```

事件回调类 SAXHandler.java

```java
import java.io.*;
import java.util.Hashtable;
import org.xml.sax.*;
public class SAXHandler extends HandlerBase
{
private Hashtable table = new Hashtable();
private String currentElement = null;
private String currentValue = null;
public void setTable(Hashtable table)
{
this.table = table;
}
public Hashtable getTable()
{
return table;
}
public void startElement(String tag, AttributeList attrs)
throws SAXException
{
currentElement = tag;
}
public void characters(char[] ch, int start, int length)
throws SAXException
```

```
{
currentValue = new String(ch, start, length);
}
public void endElement(String name) throws SAXException
{
if (currentElement.equals(name))
table.put(currentElement, currentValue);
}

}
```

JSP 内容显示源码,SaxXml.jsp:

```
<HTML>
<HEAD>
<TITLE>剖析 XML 文件 people.xml</TITLE>
</HEAD>
<BODY>
<%@ page errorPage=ErrPage.jsp
contentType=text/html;charset=GB2312 %>
<%@ page import=java.io.* %>
<%@ page import=java.util.Hashtable %>
<%@ page import=org.w3c.dom.* %>
<%@ page import=org.xml.sax.* %>
<%@ page import=javax.xml.parsers.SAXParserFactory %>
<%@ page import=javax.xml.parsers.SAXParser %>
<%@ page import=SAXHandler %>
<%
File file = new File(c:\people.xml);
FileReader reader = new FileReader(file);
Parser parser;
SAXParserFactory spf = SAXParserFactory.newInstance();
SAXParser sp = spf.newSAXParser();
```

```
SAXHandler handler = new SAXHandler();
sp.parse(new InputSource(reader), handler);
Hashtable hashTable = handler.getTable();
out.println(<TABLE BORDER=2><CAPTION>教师信息表</CAPTION>);
out.println(<TR><TD>姓名</TD> + <TD> +
(String)hashTable.get(new String(name)) +</TD></TR>);
out.println(<TR><TD>学院</TD> + <TD> +
(String)hashTable.get(newString(college))+</TD></TR>);
out.println(<TR><TD>电话</TD> + <TD> +
(String)hashTable.get(new String(telephone)) +</TD></TR>);
out.println(<TR><TD>备注</TD> + <TD> +
(String)hashTable.get(new String(notes)) +</TD></TR>);
out.println(</TABLE>);
%>
</BODY>
</HTML>
```

5、XML 文档定义有几种形式？它们之间有何本质区别？解析 XML 文档有哪几种方式？

a: 两种形式 dtd schema，b:本质区别:schema 本身是 xml 的，可以被 XML 解析器解析(这也是从 DTD 上发展 schema 的根本目的)，c:有 DOM,SAX,STAX 等

DOM:处理大型文件时其性能下降的非常厉害。这个问题是由 DOM 的树结构所造成的，这种结构占用的内存较多，而且DOM必须在解析文件之前把整个文档装入内存,适合对XML的随机访问

SAX:不现于 DOM,SAX 是事件驱动型的 XML 解析方式。它顺序读取 XML 文件，不需要一次全部装载整个文件。当遇到像文件开头，文档结束，或者标签开头与标签结束时，它会触发一个事件，用户通过在其回调事件中写入处理代码来处理 XML 文件，适合对 XML 的顺序访问

STAX:Streaming API forXML (StAX)

七.流行的框架与新技术

1、谈谈你对 Struts 的理解。

答:

1. struts 是一个按 MVC 模式设计的 Web 层框架，其实它就是一个大大的 servlet，这个 Servlet 名为 ActionServlet，或是 ActionServlet 的子类。我们可以在 web.xml 文件中将符合某种特征的所有请求交给这个 Servlet 处理，这个 Servlet 再参照一个配置文件（通常为/WEB-INF/struts-config.xml）将各个请求分别分配给不同的 action 去处理。

一个扩展知识点：struts 的配置文件可以有多个，可以按模块配置各自的配置文件，这样可以防止配置文件的过度膨胀；

2.ActionServlet 把请求交给 action 去处理之前，会将请求参数封装成一个 formbean 对象（就是一个 java 类，这个类中的每个属性对应一个请求参数），封装成一个什么样的 formbean 对象呢？看配置文件。

3.要说明的是， ActionServlet 把 formbean 对象传递给 action 的 execute 方法之前，可能会调用 formbean 的 validate 方法进行校验，只有校验通过后才将这个 formbean 对象传递给 action 的 execute 方法，否则，它将返回一个错误页面，这个错误页面由 input 属性指定，（看配置文件）作者为什么将这里命名为 input 属性，而不是 error 属性，我们后面结合实际的运行效果进行分析。

4.action 执行完后要返回显示的结果视图，这个结果视图是用一个 ActionForward 对象来表示的，actionforward 对象通过 struts-config.xml 配置文件中的配置关联到某个 jsp 页面，因为程序中使用的是在 struts-config.xml 配置文件为 jsp 页面设置的逻辑名，这样可以实现 action 程序代码与返回的 jsp 页面名称的解耦。

你对 struts 可能还有自己的应用方面的经验，那也要一并说出来。

2、谈谈你对 Hibernate 的理解。

答:

1. 面向对象设计的软件内部运行过程可以理解成就是在不断创建各种新对象、建立对象之间的关系，调用对象的方法来改变各个对象的状态和对象消亡的过程，不管程序运行的过程和操作怎么样，本质上都是要得到一个结果，程序上一个时刻和下一个时刻的运行结果的差异就表现在内存中的对象状态发生了变化。

2.为了在关机和内存空间不够的状况下，保持程序的运行状态，需要将内存中的对象状态保存到持久化设备和从持久化设备中恢复出对象的状态，通常都是保存到关系数据库来保存大量对象信息。从 Java 程序的运行功能上来讲，保存对象状态的功能相比系统运行的其他功能来说，应该是一个很不起眼的附属功能，java 采用 jdbc 来实现这个功能，这个不起眼的功能却要编写大量的代码，而做的事情仅仅是保存对象和恢复对象，并且那些大量的 jdbc

代码并没有什么技术含量，基本上是采用一套例行公事的标准代码模板来编写，是一种苦活和重复性的工作。

3.通过数据库保存 java 程序运行时产生的对象和恢复对象，其实就是实现了 java 对象与关系数据库记录的映射关系，称为 ORM（即 Object RelationMapping），人们可以通过封装 JDBC 代码来实现了这种功能，封装出来的产品称之为 ORM 框架，Hibernate 就是其中的一种流行 ORM 框架。使用 Hibernate 框架，不用写 JDBC 代码，仅仅是调用一个 save 方法，就可以将对象保存到关系数据库中，仅仅是调用一个 get 方法，就可以从数据库中加载出一个对象。

4.使用 Hibernate 的基本流程是：配置 Configuration 对象、产生 SessionFactory、创建 session 对象，启动事务，完成 CRUD 操作，提交事务，关闭 session。

5.使用 Hibernate 时，先要配置 hibernate.cfg.xml 文件，其中配置数据库连接信息和方言等，还要为每个实体配置相应的 hbm.xml 文件，hibernate.cfg.xml 文件中需要登记每个 hbm.xml 文件。

6.在应用 Hibernate 时，重点要了解 Session 的缓存原理，级联，延迟加载和 hql 查询。

3、AOP 的作用。

4、你对 Spring 的理解。

1.Spring 实现了工厂模式的工厂类（在这里有必要解释清楚什么是工厂模式），这个类名为 BeanFactory（实际上是一个接口），在程序中通常 BeanFactory 的子类 ApplicationContext。Spring 相当于一个大的工厂类，在其配置文件中通过<bean>元素配置用于创建实例对象的类名和实例对象的属性。

2. Spring 提供了对 IOC 良好支持，IOC 是一种编程思想，是一种架构艺术，利用这种思想可以很好地实现模块之间的解耦。IOC 也称为 DI（Depency Injection），什么叫依赖注入呢？

譬如，Class Programmer

```
{
        Computer computer =null;
        public void code()
        {
                //Computercomputer = new IBMComputer();
                //Computercomputer = beanfacotry.getComputer();
                computer.write();
        }
        public voidsetComputer(Computer computer)
```